UNIVERSIDAD POLITÉCNICA DE MADRID

**Escuela Técnica Superior de Ingeniería de
Sistemas Informáticos**

**ALGORÍTMICA COMPUTACIONAL BIOINSPIRADA
APLICADA A SMART CITIES: EL PARADIGMA DE
LA COMPUTACIÓN NATURAL**

TESIS DOCTORAL

Clemencio Morales Lucas

Ingeniero de Computadores

Máster en Ingeniería Web

Año 2018

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

**Algorítmica Computacional Bioinspirada aplicada a Smart Cities:
el Paradigma de la Computación Natural**

Doctorando: Clemencio Morales Lucas

**Ingeniero de Computadores
Máster en Ingeniería Web**

Directores:

Prof. Dr. Luis Fernando de Mingo López & Prof. Dra. Nuria Gómez Blas

Año 2018

Tribunal nombrado por el Magfco. y Excmo. Sr. Rector de la Universidad
Politécnica de Madrid, el día.....de.....de 20.....

Presidente:

Vocal:

Vocal:

Vocal:

Secretario:

Suplente:

Suplente:

Realizado el acto de defensa y lectura de la Tesis “**ALGORÍTMICA
COMPUTACIONAL BIOINSPIRADA APLICADA A SMART CITIES: EL
PARADIGMA DE LA COMPUTACIÓN NATURAL**”, realizada por el Doctorando
CLEMENCIO MORALES LUCAS y dirigida por los Doctores **LUIS FERNANDO
DE MINGO LÓPEZ** y **NURIA GÓMEZ BLAS** dentro del programa de doctorado
“*Doctorado en Ciencias y Tecnologías de la Computación para Smart Cities*”
el día.....de.....de 20..... en la Escuela Técnica Superior de
Ingeniería de Sistemas Informáticos.

Calificación:

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO

Quiero dedicar esta Tesis Doctoral:

A mis padres, Clemencio y Beatriz, porque gracias a su incansable esfuerzo y apoyo incondicional no sólo he llegado hasta aquí, sino que he podido escribir esta Tesis Doctoral.

A mi hermano, Carlos, por ser también mi mejor amigo.
A Puppy, que allí donde esté, a buen seguro estará orgullosa de compartir este momento conmigo, y vivirá siempre en mi memoria y en la página 298 de esta Tesis Doctoral.

A mi novia, Carolina, por estar siempre a mi lado.
A todos mis amigos y familiares, que siempre han confiado en mí.
A todos mis profesores y compañeros de la ETSISI, en especial a Luis Fernando de Mingo López y Nuria Gómez Blas, porque sin su inestimable ayuda y conocimiento, no hubiera podido realizar este proyecto.

A todos vosotros, amigos, mis más sinceros agradecimientos.

A handwritten signature in dark ink, consisting of a series of loops and a long horizontal stroke extending to the right.

Clemencio Morales Lucas

Cuando emprendas tu viaje a Ítaca
 pide que el camino sea largo,
 lleno de aventuras, lleno de experiencias.
 No temas a los lestrigones ni a los cíclopes
 ni al colérico Poseidón,
 seres tales jamás hallarás en tu camino,
 si tu pensar es elevado, si selecta
 es la emoción que toca tu espíritu y tu cuerpo.
 Ni a los lestrigones ni a los cíclopes
 ni al salvaje Poseidón encontrarás,
 si no los llevas dentro de tu alma,
 si no los yergue tu alma ante ti.
 Pide que el camino sea largo.
 Que muchas sean las mañanas de verano
 en que llegues -¡con qué placer y alegría!-
 a puertos nunca vistos antes.
 Detente en los emporios de Fenicia
 y hazte con hermosas mercancías,
 nácar y coral, ámbar y ébano
 y toda suerte de perfumes sensuales,
 cuantos más abundantes perfumes sensuales puedas.
 Ve a muchas ciudades egipcias
 a aprender, a aprender de sus sabios.
 Ten siempre a Ítaca en tu mente.
 Llegar allí es tu destino.
 Mas no apresures nunca el viaje.
 Mejor que dure muchos años
 y atracar, viejo ya, en la isla,
 enriquecido de cuanto ganaste en el camino
 sin aguantar a que Ítaca te enriquezca.
 Ítaca te brindó tan hermoso viaje.
 Sin ella no habrías emprendido el camino.
 Pero no tiene ya nada que darte.
 Aunque la halles pobre, Ítaca no te ha engañado.
 Así, sabio como te has vuelto, con tanta experiencia,
 entenderás ya qué significan las Ítacas.

"Ítaca" (1911)

Constantino Cavafis (1863 - 1933)

ÍNDICE DE CONTENIDOS

A. Resumen	19
B. Abstract	21
1. INTRODUCCIÓN Y PERSPECTIVA DE LA PROBLEMÁTICA	23
1.1 Objetivos	25
1.2 Contribución principal de la Tesis Doctoral	26
2. ESTADO DEL ARTE	29
2.1 Paradigmas de Computación Involucrados	29
2.1.1 Fireworks Algorithm	35
2.1.2 Genetic Algorithms	64
2.1.3 Ant Colony Optimization	92
2.1.4 Particle Swarm Optimization	112
2.1.5 Grammatical Evolution	139
2.1.6 Grammatical Swarm	156
2.1.7 Memetic Computing	168
2.2 Estado del arte del problema que aborda la Tesis Doctoral	177
2.2.1 Grammatical Evolution	177
2.2.2 Classic Fireworks Algorithm	180
3. ACORD ALGORITHM & IMPROVED FIREWORKS ALGORITHM. METODOLOGÍA	183
3.1 Ant Colony Optimization Rule Derivation Algorithm (ACORD)	183
3.1.1 Definición y Características	186
3.1.2 Resultados con Santa Fe Trail	193
3.2 Improved Fireworks Algorithm	202
3.2.1 Explosión Gaussiana de Sparks	204
3.2.2 <i>Divide & Conquer, Loop Unrolling</i> y Decremento de la Complejidad Algorítmica	213
3.2.3 Mejora de la complejidad espacial y datos aleatorios	219
3.2.4 Diagrama de Flujo	220
3.3 Metodología	222
3.3.1 Condiciones y parámetros de la comparativa	223

3.3.2 <i>Benchmarking</i> y Métricas de Rendimiento	224
3.3.2.1 Función Ackley	224
3.3.2.2 Función Schwefel	225
3.3.2.3 Función Sphere	226
3.3.2.4 Función Rosenbrock	226
3.3.2.5 Función Griewanks	227
3.3.2.6 Función Rastrigin	228
3.3.2.7 Función Goldstein	229
3.3.2.8 Función Schaffer	229
3.3.2.9 Función Rotated Hyper Ellipsoid (Sum Squares)	230
3.3.3 Comparativa entre Funciones de Benchmarking	231
4. RESULTADOS	233
5. ANÁLISIS DE LOS RESULTADOS Y DISCUSIÓN	238
6. LÍNEAS FUTURAS DE INVESTIGACIÓN	245
6.1 Esquema Híbrido entre FA & GA	245
6.2 Esquema Híbrido entre FA & ACO	248
6.3 Mejora progresiva de sub-secciones de FA	251
7-A. CONCLUSIONES	255
7-B. CONCLUSIONS	260
Referencias Bibliográficas	264
Anexo I: Tabla de Objetivos contra Resultados de la Tesis Doctoral	286
Anexo II: Publicaciones relacionadas con la Tesis Doctoral	288
Anexo III: Líneas Prácticas de Investigación y Aplicación a Smart Cities	294
A.1 Líneas Prácticas	294
A.1.1 Sistema Enjambre	294
A.1.2 Aplicación basada en Híbrido FA-ACO para zonas rurales	297
A.2 Aplicación a Smart Cities	302
A.2.1 Smart Cities	303
A.2.1.1 Ejemplos de Smart Cities	311
A.2.1.2 Justificación	314
A.2.1.3 Control de Masas	318
A.2.1.4 Análisis de las alternativas de resolución existentes	325

Anexo IV: Artículo JCR Q2 Publicado en <i>Sensors</i>: Natural Computing Applied to the Underground System: A Synergistic Approach for Smart Cities	338
Anexo V: Artículo JCR Q2 Publicado en <i>Soft Computing</i>: Ant Colony systems Optimization applied to BNF grammars Rule Derivation (ACORD Algorithm)	360
Anexo VI: Artículo en trámite de publicación Q2: Pattern Recognition with Convolutional Neural Networks: Humpback Whale Tails	377

ÍNDICE DE FIGURAS

Figura I: Antena Evolucionada del vehículo espacial <i>ST5</i> , concebida con Algorítmica Natural	30
Figura II: Explosión de Fuegos Artificiales	36
Figura III: Símil entre la explosión de los Fuegos Artificiales y <i>Fireworks Algorithm</i>	37
Figura IV: Diagrama de Flujo de un FA	39
Figura V: Explosión óptima de un <i>firework</i> en (a) y homóloga de baja calidad en (b)	41
Figura VI: Algoritmo de generación de <i>sparks</i> en FA	42
Figura VII: Mutación Gaussiana en el espectro del FA	43
Figura VIII: Algoritmo de Mutación Gaussiana en FA	43
Figura IX: Modelo de <i>Standard Particle Swarm Optimization</i> : Actualización posicional a través de una topología estrella (<i>gBest</i>) y un término inercial, <i>w</i>	46
Figura X: Resultados sobre las Funciones de <i>Benchmarking</i> fruto de la comparación entre FA, CPSO y SPSO	46
Figura XI: Media Estadística y Desviación Típica de las soluciones encontradas por FA, CPSO y SPSO en nueve Funciones de <i>Benchmarking</i> sobre 20 ejecuciones independientes de 10000 evaluaciones de función	47
Figura XII: Diagrama de Bloque de un controlador Proporcional, Integral y Derivativo (PID)	48
Figura XIII: Resultado de la ejecución del <i>Fireworks Algorithm</i> clásico implementado en la presente Tesis Doctoral	61
Figura XIV: <i>Teoría de la Jirafa</i> de Jean Baptiste Lamarck	64
Figura XV: Cruzamiento Monohíbrido Mendeliano entre dos <i>Pisum Sativum</i>	66
Figura XVI: Esquema de la técnica de Cruzamiento o <i>Crossover</i> en un punto	69
Figura XVII: Esquema de la técnica de Cruzamiento o <i>Crossover</i> en dos puntos	69
Figura XVIII: Esquema de la técnica de Cruzamiento o <i>Crossover corte y empalme</i>	69
Figura XIX: Esquema de la técnica de Cruzamiento o <i>Crossover</i> uniforme	70
Figura XX: Esquema de la técnica de Cruzamiento o <i>Crossover</i> uniforme medio	70
Figura XXI: Estructura atómica de un cromosoma	71
Figura XXII: Funcionamiento genérico esquematizado de un Algoritmo Genético	72
Figura XXIII: Diagrama de flujo genérico de un Algoritmo Genético	73
Figura XXIV: Fotograma de <i>Quake III Arena</i>	74
Figura XXV: Problemática de los GA en la búsqueda de óptimos globales en el espacio de búsqueda	75
Figura XXVI: Tabla de conversión entre el Sistema Decimal, Binario y <i>Gray</i>	79
Figura XXVII: Resultado de la ejecución del GA representado en <i>UltimateMenaceCreator.java</i>	88
Figura XXVIII: Figura geométrica compuesta por una bandada de <i>Columba Livia</i> en base a la Inteligencia de Enjambre	93
Figura XXIX: Ejemplo esquemático ilustrativo del Experimento de Goss	94
Figura XXX: Ejemplificación de la entidad matemática: Grafo no dirigido ponderado	96
Figura XXXI: Correlación entre los conceptos del mundo natural y el computacional en el ámbito del ACO	97
Figura XXXII: Representación esquemática del Algoritmo de la Colonia de Hormigas	97
Figura XXXIII: Representación gráfica del Algoritmo de la Colonia de Hormigas	100
Figura XXXIV: Comparativa de rendimiento entre el Algoritmo de la Colonia de Hormigas y sus principales variantes	108
Figura XXXV: Banco de <i>Thunnus Albacres</i> reagrupándose frente a una hipotética amenaza	113

Figura XXXVI: Hipersuperficie ilustrativa del rendimiento de PSO ante un <i>benchmark</i> de problemas biparamétricos	119
Figura XXXVII: Topologías de red susceptibles de aplicación al enjambre adscrito al algoritmo PSO	123
Figura XXXVIII Topología en estrella (izq.) y en anillo (der.)	123
Figura XXXIX: Cepas de producción de microorganismos de potenciación del crecimiento del <i>Olea Europaea</i>	128
Figura XL: Resultado de la ejecución del PSO representado en <i>ParticleSwarmOptimization.java</i>	136
Figura XLI: Directrices BNF para los identificadores de variable en Java	140
Figura XLII: Analogías en el proceso evolutivo entre la GE y un sistema biológico clásico	144
Figura XLIII: <i>Screenshot</i> de carga correcta de la biblioteca <i>GEVA v2.0</i> para algorítmica GE	150
Figura XLIV: <i>Screenshot</i> de la GUI asociada al software <i>GEVA v2.0</i>	150
Figura XLV: <i>Screenshot</i> asociado a la ejecución de <i>HelloWorld.properties</i> en <i>GEVA v2.0</i>	153
Figura XLVI: <i>Screenshot</i> asociado a la sección Console de <i>HelloWorld.properties</i> en <i>GEVA v2.0</i>	154
Figura XLVII: Formación de una bandada de <i>Leucophaeus Scoresbii</i> frente a la formación militar en orbe	156
Figura XLVIII: Flujograma adscrito al funcionamiento del Algoritmo <i>Grammatical Swarm</i>	159
Figura XLIX: Escenario adscrito a la matriz toroidal representativa del problema <i>Santa Fe Ant Trail</i>	161
Figura L: Gráfica adscrita a la aptitud media en el problema <i>Santa Fe Trail</i> (izq) y frecuencia	162
Figura LI: Esquema lógico de un multiplexor 2 a 1 con una entrada de selección y del equivalente conmutador controlado	163
Figura LII: Gráfica adscrita a la aptitud media en el Problema del Multiplexor	164
Figura LIII: Resultados arrojados tras la comparativa empírica entre el algoritmo GS y su competidor GE	165
Figura LIV: Diagrama de Flujo de un Algoritmo Memético	171
Figura LV: Áreas representando el volumen adscrito al <i>Napkin Ring Problem</i>	174
Figura LVI: Método principal para la resolución de <i>Napkin Ring Problem</i> con <i>jMeme</i>	175
Figura LVII: ACO resolviendo TSP al hallar una ruta mínima entre los puntos de interés	185
Figura LVIII Primera iteración de ACORD	188
Figura LIX: Segunda iteración de ACORD	189
Figura LX: Gramática de entrada y resultado obtenido con un RRDS=15	190
Figura LXI: Gramática de entrada (II) y resultado obtenido con un RRDS=15	191
Figura LXII: Diagrama asociado al problema Santa Fe Trail en un grid toroidal de 32x32 celdas	194
Figura LXIII: Gramática BNF para Santa FE para estudiar el impacto de la localidad en GE	196
Figura LXIV: Fitness vs # de iteración en Santa Fe Trail problem usando ACORD con la gramática Koza	201
Figura LXV: Histograma de la distribución Gaussiana	205
Figura LXVI: Conversión de varias distribuciones probabilísticas hacia la Distribución Gaussiana conforme N crece (Teorema Central del Límite)	207
Figura LXVII: Salto de complejidad cuadrática a $n \log--star n$ en la Pirámide de Complejidades Comunes	219
Figura LXVIII: Diagrama de Flujo de un <i>Fireworks Algorithm</i> (FA)	220
Figura LXIX: Diagrama de Flujo de un <i>Improved Fireworks Algorithm</i> (IFA)	222
Figura LXX: Función <i>Ackley</i> - Representación gráfica bidimensional	225
Figura LXXI: Función <i>Schwefel</i> - Representación gráfica bidimensional	225
Figura LXXII: Función <i>Sphere</i> - Representación gráfica bidimensional	226
Figura LXXIII: Función <i>Rosenbrock</i> - Representación gráfica bidimensional	227

Figura LXXIV: Función <i>Griewanks</i> - Representación gráfica bidimensional	228
Figura LXXV: Función <i>Rastrigin</i> - Representación gráfica bidimensional	228
Figura LXXVI: Función <i>Goldstein</i> - Representación gráfica bidimensional	229
Figura LXXVII: Función <i>Schaffer</i> - Representación gráfica bidimensional	230
Figura LXXVIII: Función <i>Rotated Hyper Ellipsoid</i> - Representación gráfica bidimensional	231
Figura LXXIX: Modelo estándar de PSO; actualización posicional utilizando una topología en estrella (<i>gBest</i>) y un término inercial, <i>w</i>	247
Figura LXXX: POI (<i>Points of Interest</i>) encerrados en un área de búsqueda	249
Figura LXXXI: Representación del FA y <i>sparks</i> susceptibles de hallar POI (<i>Points of Interest</i>)	250
Figura LXXXII: Esbozo de la propuesta híbrida entre FA/ACO	251
Figura LXXXIII: Diagrama de Flujo de FA (izquierda) y su contraparte en IFA (derecha)	252
Figura LXXXIV: Fotografía esquemática del sistema de componentes de la <i>Wireless Sensor Network</i>	295
Figura LXXXV: Diseño esquemático del sistema enjambre propuesto	296
Figura LXXXVI: Imagen ejemplificativa del Centro de Control y Operaciones	297
Figura LXXXVII: Puppy - Imagen de entrada para el repositorio <i>TensorFlow</i>	298
Figura LXXXVIII: Métodos de <i>mapping</i> , reducción y normalización en el código fuente del ejemplo <i>TensorFlow</i>	299
Figura LXXXIX: Resultado del <i>software</i> al analizar la imagen de Puppy (<i>Canis Lupus Familiaris</i>)	299
Figura XC: <i>Gran Tiburón Blanco</i> - Imagen de entrada para el repositorio <i>TensorFlow</i>	300
Figura XCI: Resultado del <i>software</i> al analizar el <i>Gran Tiburón Blanco</i> (<i>Carcharodon Carcharias</i>)	300
Figura XCII: Imagen tipo, susceptible de constituir el <i>output</i> de la flota de drones	301
Figura XCIII: Diseño esquemático del sistema de rescate basado en FA y la flota de drones	302
Figura XCIV: Datos susceptibles de monitorización en una <i>smart city</i>	303
Figura XCV: Diagrama de <i>Venn</i> ejemplificativo de los compromisos de una <i>smart city</i>	308
Figura XCVI: Panorámica de Río de Janeiro tras la inversión orientada a la consecución de una <i>smart city</i>	315
Figura XCVII: Interior del estadio Metropolitano	318
Figura XCVIII: Estación de la Línea 7 del Sistema de Metro de Madrid; Estadio Olímpico	319
Figura XCIX: Comparativa de vehículos utilizados en el transporte de 200 personas	323
Figura C: Diagrama de sectores adscrito al perfil de los usuarios de STPU por edad y sexo	323
Figura CI: Ejemplo de detección de situaciones de índole problemática por el sistema <i>FLIR</i>	327
Figura CII: <i>Screenshot</i> del menú principal asociado a la aplicación <i>Metro de Madrid Oficial</i>	328
Figura CIII: <i>Screenshot</i> de la sección <i>Líneas y Estaciones</i> de la aplicación <i>Metro de Madrid Oficial</i>	329
Figura CIV: <i>Screenshot</i> de la sección <i>Trayectos</i> de la aplicación <i>Metro de Madrid Oficial</i>	330
Figura CV: <i>Screenshot</i> de la sección <i>Opciones</i> (de Trayecto) de la aplicación <i>Metro de Madrid Oficial</i>	331
Figura CVI: <i>Screenshot</i> del trayecto a recorrer calculado por la aplicación <i>Metro de Madrid Oficial</i>	332

ÍNDICE DE ALGORITMOS

Algoritmo I: Ant Colony Algorithm for BNF Grammars	187
Algoritmo II: Estructura general de FA	204
Algoritmo III: Generación de un <i>explosion spark</i> en IFA	209
Algoritmo IV: Generación de un <i>Gaussian spark</i> en IFA	209
Algoritmo V: Estrategia de actualización de amplitud de la explosión en <i>dynFA</i>	210
Algoritmo VI: Estrategia de actualización de amplitud de la explosión en AFA	213
Algoritmo II (2): Estructura general de FA	221

ÍNDICE DE TABLAS

Tabla I: Tabla de inputs para el experimento contra el Algoritmo ACORD	197
Tabla II: Resultados experimentales para Santa Fe Trail aplicado al Algoritmo ACORD	199
Tabla III: Resultados experimentales para Santa Fe Trail aplicado a Grammatical Evolution usando BNF-Koza	199
Tabla IV: Resultados experimentales para Santa Fe Trail aplicado a Grammatical Evolution usando BNF-O'Neill	199
Tabla V: Tiempo de ejecución (s) para cada par <i>Benchmark function-Shift</i> y tiempo total necesitado para FA	234
Tabla VI: Tiempo de ejecución (s) para cada par <i>Benchmark function-Shift</i> y tiempo total necesitado para IFA	235
Tabla VII: Tiempo total de ejecución (s) para cada par <i>Benchmark function-Shift</i> y tiempo total necesitado para FA e IFA	236
Tabla VIII: Tiempo de mejora (s) para cada par <i>Benchmark function-Shift</i> al restar FA-IFA y porcentaje total de mejora entre FA e IFA por la <i>Ley de Amdahl</i>	241
Tabla IX: Tabla comparativa entre <i>running length</i> y <i>proportion of functions</i> de varios algoritmos evolutivos de optimización	243
Tabla X: Tabla de Objetivos contra Resultados de la Tesis Doctoral	286
Tabla XI: Tabla de Objetivos del Programa de Doctorado en Ciencias y Tecnologías de la Computación	292

A. Resumen

El propósito de la presente Tesis Doctoral se fundamenta en, una vez otorgada al lector una visión teórica exhaustiva, fruto de la investigación en el estado del arte e idiosincrasia de los Paradigmas de Computación adscritos a la Computación Natural, como Fireworks Algorithm, Genetic Algorithm, Ant Colony Optimization, Grammatical Evolution, Grammatical Swarm o Particle Swarm Optimization entre otros, acometer la mejora algorítmica empíricamente constatable de una de estas entidades bioinspiradas, más en concreto Fireworks Algorithm, dando lugar al nacimiento de Improved Fireworks Algorithm (IFA). Por otro lado, se presentará una novedosa sinergia entre un importante grupo de entidades del anteriormente citado conjunto; Ant Colony Optimization, Grammatical Evolution, Grammatical Swarm y Genetic Algorithms dando lugar a la gestación del novedoso algoritmo genérico ACORD (**Ant Colony systems Optimization applied to BNF grammars Rule Derivation**), que utiliza dichas técnicas algorítmicas para explorar gramáticas en nomenclatura Backus-Naur Form (BNF) cuyos elementos se asocian con soluciones a un problema dado. Por tanto, si bien la mejora de Fireworks Algorithm es de índole numérica, ACORD representa una creación conceptual, demostrando empíricamente la mastodóntica potencia del binomio teórico-práctico que brinda el Paradigma de la Computación Natural.

Tras hallar tal enriquecimiento o mejora a nivel algorítmico con IFA, así como la sinergia entre las citadas entidades bajo el espectro de la Computación Natural con ACORD, se describirá pormenorizadamente la metodología aplicada durante el logro de dichos hitos, en tanto en cuanto qué se ha mejorado y cómo se ha logrado. De manera previa a su análisis se esgrimirán los resultados obtenidos al finalizar los experimentos pertinentes, así como un análisis y discusión de los resultados obtenidos a partir de dichos experimentos. Es reseñable que, entre otras publicaciones, la base teórica sobre Natural Computing ha valido la publicación de [Morales Lucas et al, 2018] (JCR Q2). En la misma línea, ACORD ha merecido la publicación de [de Mingo et al, 2019] (JCR Q2).

Finalmente, una vez lograda la consecución de las metas anteriormente expuestas, se expondrán las líneas futuras de investigación abiertas por la presente Tesis Doctoral, así como las conclusiones que de la misma se extraen, acompañadas de los anexos pertinentes para complementar la información necesaria.

Palabras clave: Computación Natural, Fireworks Algorithm, Genetic Algorithm, Ant Colony Optimization, Grammatical Evolution, Grammatical Swarm, Particle Swarm Optimization, Backus-Naur Form.

B. Abstract

This Ph.D purpose is, once a thorough theoretical research is given to the reader, and as a result of the investigation in the state of the art and idiosyncrasy regarding the Computing Paradigms belonging Natural Computing, like Fireworks Algorithm, Genetic Algorithm, Ant Colony Optimization, Grammatical Evolution, Grammatical Swarm or Particle Swarm Optimization, among others, to face the empirically-observable algorithmic improvement of one of the aforementioned bio-inspired entities, more concretely Fireworks Algorithm, giving birth to Improved Fireworks Algorithm (IFA). Furthermore, a novel synergy among four important elements within the aforementioned set will be presented; Ant Colony Optimization, Grammatical Evolution, Grammatical Swarm and Genetic Algorithms, what disembogues in the creation of the brand new, generic ACORD Algorithm (**Ant Colony systems Optimization applied to BNF grammars Rule Derivation**), which merges these paradigms in order to explore Backus-Naur Form (BNF) grammars, whose elements represent solutions to a given problem. Thus, should Fireworks Algorithm improvement aligns with a numerical idiosyncrasy, ACORD represents a conceptual creation, empirically demonstrating the colossal effectiveness of the theoretical-practical binomial that Natural Computing hands over.

Once that algorithm enrichment has been achieved with IFA, as well as the synergy among entities has been accomplished under Natural Computing spectrum with ACORD, the applied methodology to reach that goal is to be described with the finest granularity level. Hence, elucidating what has been improved and how. All results thrown by means of the pertinent experiments will be shown, as well as its opportune analysis and discussion. It is noteworthy that, among other publications, the theoretical basis on Natural Computing have deserved the publication of [Morales Lucas et al, 2018] (JCR Q2). Along the same lines, ACORD has deserved the publication of [de Mingo et al, 2019] (JCR Q2).

Finally, once the achievement of the aforementioned goals has been accomplished, the future investigation paths lead by this Ph.D, accompanied with the applicable conclusions will be shown, reinforced with the relevant annexes in order to complement the needed information.

Keywords: Natural Computing, Fireworks Algorithm, Genetic Algorithm, Ant Colony Optimization, Grammatical Evolution, Grammatical Swarm, Particle Swarm Optimization, Backus-Naur Form.

1. INTRODUCCIÓN Y PERSPECTIVA DE LA PROBLEMÁTICA

La síntesis entre la idiosincrasia del espectro más puramente informático con el atómicamente natural, puede haber escrito los primeros versos del Libro del Apocalipsis de la Computación Clásica, dando lugar a una concepción radicalmente renovada de la misma.

La naturaleza, entendiéndose como entidad, realiza sus funciones de una manera que es tan apasionante como aún desconocida en su mayoría para la comunidad científica. Más allá de los fenómenos visibles que acontecen bajo su manto, existen innumerables factores, frecuentemente invisibles. No son pocos los filósofos y científicos que han observado e investigado estos fenómenos desde hace siglos, tratando de entender, explicar, adoptar, adaptar y replicar los mismos en los sistemas de origen artificial. Tanto en el mundo de los seres vivos como en su homólogo inerte, existe un mastodóntico número de agentes y fuerzas, de los cuales un gran porcentaje es aún desconocido para el ser humano. Muchos de ellos, podrían estar impregnados de una complejidad subyacente que va más allá del potencial de la comprensión humana. Tales agentes actúan en paralelo, y muchas veces enfrentándose entre sí, dando forma a la naturaleza y regulando la armonía que la caracteriza.

La anteriormente citada característica puede apreciarse como un dialecto de la naturaleza, que subyace en el concepto de la evolución asociada al mundo natural. La evolución de la complejidad en el dicho mundo sigue un orden distintivo, donde el procesamiento de la información se lleva a cabo de una manera distribuida, no centralizada y óptima, sin la necesidad de ningún ente que aúne el control de los numerosos procesos que en ella acontecen.

Para ello, la naturaleza se distribuye en un amplio conjunto de elementos que bien parecen seguir un enfoque *bottom-up*, donde los más simples organismos unicelulares y pequeñas moléculas se unen dando lugar a complejas estructuras macromoleculares formadas a partir de la asociación de subunidades proteicas definidas. Secuencias como la anteriormente citada como ejemplo, expresan la dependencia mutua y relación, en términos de estructura, de los agentes presentes en la naturaleza. Todos estos fenómenos, bien sean parcial o totalmente conocidos, emergen dando lugar a nuevos campos de la ciencia y la tecnología, en los que se pretende conseguir que simples agentes colaboren (o compitan) para hallar la solución a problemas cuya índole parece solicitar, de manera subyacente, un cambio paradigmático y de concepción.

Tras recortar las numerosas capas de complejidad que la envuelven, es lícito entender, por ende, la ciencia como un diálogo entre la comunidad científica y la naturaleza [Prigogine et al., 1996], que ha evolucionado a lo largo de los siglos, dando lugar a nuevos conceptos, métodos y herramientas aplicadas a disciplinas de la ciencia y la ingeniería.

Es, en este punto, pertinente definir la Computación Natural como el campo de investigación que toma por objetivo tanto la computación clásica como la inspirada en la naturaleza [Rozenberg et al., 2012], no sólo persiguiendo sinergias, sino investigando a su vez, en términos de procesamiento de la información, todos aquellos fenómenos acontecidos en el entorno natural.

Dentro de la Computación Natural, es posible encontrar ejemplos de aplicaciones que varían en un amplio rango: Autómatas Celulares inspirados en la comunicación intercelular [Crols et al., 2017], Computación Neuronal inspirada en la evolución Darwiniana [Koza, 92], Inteligencias de Enjambre inspiradas por el comportamiento de grupos de organismos [Blum et al., 2008], Sistemas Inmunes Artificiales basados en su homólogo inmunológico humano [de Castro et al., 2002], Computación de Membranas [Stefan, 2002], Morfogénesis aplicada a la Computación [Bard, 1992], y un largo etcétera que refuerza la sensación subyacente de que la Computación Natural está, casi únicamente, acotada por la imaginación y el raciocinio humano.

Huelga, por los argumentos anteriormente presentados, reseñar que la comunidad científica se erige como un testigo privilegiado de tan excitantes interacciones entre las Ciencias de la Computación y las Ciencias Naturales. Mientras que ésta absorbe ágilmente nociones, técnicas, y metodologías intrínsecas al procesamiento de la información, aquella se adapta extendiendo su noción tradicional de computación, así como sus técnicas, para expandir la computación con aquellos fenómenos naturales que acontecen en el mundo analógico. De esta forma, la Computación Natural no sólo debe verse como un aglutinante de estas interacciones, sino como un potente catalizador a su vez.

Como el avezado lector a buen seguro intuye, tan eminente paradigma no debe estancarse, siendo mejorado e hibridado para permitir que su evolución acontezca a un ritmo directamente proporcional a la de los retos que enfrenta.

A fin de facilitar dicha tarea, se introduce la presente Tesis Doctoral, cuyos objetivos, como se detallará en el posterior epígrafe, se basan en la mejora y/o consecución de entidades algorítmicas híbridas que nutran y apoyen el transcendental Paradigma de la Computación Natural.

1.1 Objetivos

Persiguiendo la finalidad de facilitar el seguimiento de la presente Tesis Doctoral, los objetivos a alcanzar en el ámbito de la misma han sido escindidos en una terna teórico-práctica, que se presenta seguidamente:

[OBJ-1]: Ofrecer una exhaustiva visión panorámica del estado del arte del Paradigma de la Computación Natural, constituyendo una base teórica para la Tesis Doctoral. Durante dicho proceso, se introducirá la problemática global y su perspectiva. Durante su desarrollo, se velará por la introducción de los posibles nuevos paradigmas adscritos al espectro de la Computación Natural y Bioinspirada que puedan ver la luz, manteniendo un estado del arte de rabiosa actualidad. Asimismo, en los casos oportunos, se definirá un repositorio *software* de ejemplos de aplicación para cada metodología algorítmica adscrita al paradigma objeto de estudio.

[OBJ-2]: Alcanzar una sinergia entre *Ant Colony Optimization* (ACO) y *Grammatical Evolution* (GE). Emplear dicha sinergia partiendo de la hipótesis de usar ACO para explorar una gramática BNF cuyos elementos son la solución a un problema dado. Implementar la sinergia entre ACO y GE a nivel de *software*.

[OBJ-3]: Mejorar empíricamente (en términos de rendimiento) *Fireworks Algorithm* (FA), partiendo de la hipótesis de que podría variarse a nivel conceptual para lograr un mejor desempeño gracias a la aplicación de las técnicas *Divide & Conquer*, *Loop Unrolling* y un enfoque Gaussiano en la explosión de *sparks*. Implementar los algoritmos resultantes a nivel de *software*.

Es reseñable que [OBJ-3] implica el diseño e implementación de dicho algoritmo en su versión clásica, lo idéntico a su vez para su versión mejorada, el conjunto de *benchmarks* utilizados en la comparación cuantitativa de los mismos y el marco *software* que englobe las entidades citadas, de forma que su mejora pueda demostrarse empíricamente.

1.2 Contribución principal de la Tesis Doctoral

La presente Tesis Doctoral contiene, además de la base teórica exhaustiva necesaria para su comprensión, el trabajo relacionado con la implementación, análisis y aplicación de los nuevos algoritmos ACORD (*Ant Colony systems Optimization applied to BNF grammars Rule Derivation*) e IFA (*Improved Fireworks Algorithm*). A lo largo del documento, se describen las premisas que desembocan en el nacimiento de sendos nuevos algoritmos, la implementación algorítmica consiguiente y el análisis de los mismos. En adición, tanto ACORD como IFA son probados contra novedosos algoritmos en cada uno de sus campos, a fin de cuantificar su potencia computacional.

De manera sucinta, las principales contribuciones de la presente Tesis Doctoral se enumeran a continuación:

- Creación del novedoso algoritmo ACORD, que abre una nueva rama adscrita a *Swarm Computing* en forma de híbrido entre *Genetic Algorithms*, *Ant Colony Optimization* y *Grammatical Evolution*. Dicho algoritmo es genérico y capaz de adaptarse a cualquier problema de *pathfinding* cambiante, resolviendo cualquier camino perseguido. El rendimiento del Algoritmo ACORD ha valido su publicación en el artículo JCR Q2 [de Mingo et al, 2019].
- Partiendo de la premisa de utilizar *Ant Colony Optimization* (ACO) para explorar una gramática BNF cuyos elementos son la solución a un problema dado, ACORD consigue un 90% de éxito medio frente al 78% y 10% de sus competidores, *Grammatical Evolution* con gramática O'Neil y *Grammatical Evolution* con gramática Koza, respectivamente. Es reseñable que, hasta la fecha, no existía ningún modelo similar que utilizara ACO.
- La base teórica fruto de la investigación sobre el Paradigma de la Computación Natural en la presente Tesis Doctoral, unida a una novedosa aplicación para Smart Cities ha valido su publicación en el artículo JCR Q2 [Morales Lucas et al, 2018].
- Concepción de un nuevo algoritmo tomando *Fireworks Algorithm* (FA) como base, denominado *Improved Fireworks Algorithm* (IFA), en el que se rediseñan casi por completo ciertas áreas del algoritmo. Dada la juventud de FA (razón por la que la mayoría de estudios al respecto se basan en las aplicaciones conceptuales del mismo) se parte de la premisa de si el algoritmo puede mejorarse, logrando un mejor rendimiento. Dicha hipótesis, que prueba ser cierta *a posteriori*, parte de la intuición de que la

aplicación de las técnicas *Divide & Conquer*, *Loop Unrolling* y un enfoque Gaussiano en la explosión de *sparks* puede mejorar el desempeño del algoritmo.

- Partiendo de dicha hipótesis, se consigue demostrar empíricamente que IFA obtiene un rendimiento del 42,9% mejor que FA. Hasta el momento, ninguna investigación se había centrado en la mejora del *throughput* (rendimiento) algorítmico en FA.

2. ESTADO DEL ARTE

Persiguiendo la finalidad de introducir al lector en la vanguardia de los Paradigmas de Computación y algoritmos inmiscuidos en la presente Tesis Doctoral, el presente epígrafe tiene como objetivo el de describir tales entidades algorítmicas.

Posteriormente, una vez relatadas dichas técnicas con la mayor profundidad posible, se aplicará el estado del arte especificado a la problemática planteada por el presente proyecto. De esta forma, se perseguirá la sinergia algorítmica entre dichas entidades, o la mejora en términos de *throughput* o rendimiento de alguna de ellas.

2.1 Paradigmas de Computación Involucrados

A pesar de su aparente disparidad, los Paradigmas de Computación y las técnicas algorítmicas a continuación descritas se enmarcan, en mayor o menor medida, en el Paradigma de la Computación Natural.

Como puede intuirse, las entidades algorítmicas adscritas a dicho paradigma tienen, como principal base, la lógica adscrita a fenómenos presentes en la naturaleza, así como la relacionada con la base genético-molecular de los seres vivos y la evolución natural, por ende. De manera formal, se denomina Computación Natural [Rozenberg, 2012] al conjunto de métodos informáticos que se circunscriben a, al menos, una característica perteneciente a la siguiente terna:

1. Obtienen su base partiendo de la observación de la naturaleza, tratando de establecer un símil a nivel informático.
2. Fundamentan su razonamiento en el uso de los computadores para sintetizar fenómenos naturales.
3. Utilizan materiales de corte natural, tales como cadenas de ADN o cromosomas, para llevar a cabo sus procesos computacionales.

Una vez descritas las características someras de la Computación Natural, es reseñable que sus principales campos de aplicación son, típicamente, las Redes Neuronales Artificiales (Artificial Neural Networks), los Algoritmos Evolutivos (Evolutionary Algorithms) [Olarius et al., 2005], la Inteligencia en Enjambre (Swarm Intelligence), la Geometría Fractal (Fractal Geometry) y la Computación de ADN (DNA Computing) [Yang, 2008], por citar algunos. Los campos de aplicación de la Computación Natural están, casi únicamente, acotados por la

imaginación. Baste como refuerzo de la anterior aseveración la siguiente imagen, cortesía de la Administración Nacional de la Aeronáutica y del Espacio estadounidense (NASA; National Aeronautics and Space Administration), en la que se aprecia la antena del vehículo espacial ST5.

Esta especial antena, conocida como antena evolucionada, toma su nombre a partir de su diseño, basado en un software de simulación fundamentado en la Computación Natural que mimetiza la Evolución Darwiniana en base a un Algoritmo Evolutivo [Hornby et al., 1999]. El aparentemente enrevesado aspecto de la antena, delimitado por el software anteriormente citado, tiene su razón de ser en la maximización del patrón de radiación de la misma. Para dar con dicho patrón, el software parte de diseños simplificados de antenas al azar, modificando elementos de las mismas en base a una secuencia semi-aleatoria que pretende dar lugar a un nuevo elenco de antenas candidatas. Tales antenas serán evaluadas para determinar en qué cuantía satisfacen los requisitos de radiación, recibiendo una calificación asignada a la finalización del proceso. El anteriormente citado algoritmo es repetido un elevado número de ocasiones, hasta que el incremento en la puntuación de las mejores antenas varía en poco o ningún valor. La antena con una mayor posición en el ranking será elegida como candidata final, mientras que el resto serán descartadas. Los algoritmos adscritos a este principio (denominados Algoritmos Genéticos) serán tratados con profundidad en el epígrafe 2.1.2 *Genetic Algorithms*, al que se referencia al lector para una mayor información. La siguiente figura ilustra el diseño final de la antena del vehículo espacial ST5:

Figura I: Antena Evolucionada del vehículo espacial ST5, concebida con Algorítmica Natural

Por otro lado, en cuanto a los paradigmas computacionales estudiados por la Computación Natural se refiere, se destaca que dichos paradigmas constituyen

abstracciones de fenómenos naturales tan diversos como la reproducción (sexual y asexual), el funcionamiento interno de una colonia de hormigas, el funcionamiento mismo del cerebro humano, la evolución Darwiniana, el comportamiento gregario, la lógica adscrita al sistema inmunológico y la optimización de los patrones de radiación de antenas, como se ha visto anteriormente. A su vez, el ámbito de plataformas donde la Computación Natural puede llevarse a cabo no sólo se limita al ámbito puramente informático, puesto que los Algoritmos Naturales pueden implementarse a su vez en medios físicos alternativos, tales como biomoléculas de ADN o ARN.

Aplicando la lógica adscrita a la Computación Natural, los procesos que acontecen en la naturaleza pueden observarse como un simple procesamiento de información clásico, en el que unos determinados inputs o entradas actúan condicionados por su entorno, generando ciertos outputs o salidas [de Castro, 2007]. Es digno de mención que, dada la disparidad de los distintos problemas que pueden resolverse mediante la Computación Natural, el esfuerzo requerido para extrapolar la base de tan dispares problemas se antoja extremadamente alto. En efecto, autores como Zuse-Fredkin establecen la tesis, en 1960, de que el universo en su completitud constituye una suerte de autómata celular gigantesco, que continuamente actualiza las directrices en base a las cuales opera. Actualmente, parafraseando la tesis de Zuse-Fredkin, se sospecha que el universo podría ser una suerte de ordenador cuántico que programa su propio comportamiento.

Habiendo introducido al lector previamente en los fundamentos básicos de la Computación Natural, a continuación se relatarán los distintos Modelos de Computación inspirados en la naturaleza existentes en la actualidad [Klug et al., 2000]:

- **Autómatas Celulares:** Se denominan como tal a todos aquellos sistemas dinámicos consistentes en un vector de células. Dado que las magnitudes espacio temporales constan de un carácter discreto desde el punto de vista matemático, cada célula (comprendida en un punto del espacio durante un instante de tiempo) podrá contar con un número de estados finitos. El autómata celular actualiza el estado de sus diferentes células de forma síncrona en función de las reglas de transiciones establecidas previamente. El siguiente estado para cada célula se computa llevando a cabo una directriz de transición, la cual depende no sólo de su estado, sino de su entorno a su vez, siendo este conformado por las células vecinas. El Juego de la Vida de Conway, símbolo hacker internacional, es uno de los mejores ejemplos de autómata celular, que a su vez es computacionalmente universal. Los autómatas celulares han sido extensivamente aplicados al modelado de una gran variedad de escenarios, entre los que destacan aquellos que simulan el ciclo vital de los seres vivos.

- **Computación Neuronal:** Se conoce como tal a la rama científica que nace de la comparación empírica entre los computadores y el sistema nervioso del ser humano.

Es reseñable que la Computación Neuronal busca alcanzar una doble meta; por un lado, esta ciencia pretende comprender cómo funciona el cerebro de un ser vivo, mientras que por otro lado busca diseñar algoritmos eficientes fundamentados en los principios según los cuales el cerebro procesa la información que recoge a través de los sentidos. Con el fin de lograr dicha dupla de metas, se recurre a las denominadas Redes Neuronales Artificiales (Artificial Neural Networks). En dichas redes, una neurona artificial ejecuta una función que recibe cierta información de entrada, con unos respectivos valores de peso y produciendo una salida acorde, que se circunscribe una función aritmética. Ciertas neuronas son seleccionadas para llevar a cabo la función de salida de información, mientras que la función de red, de tipo vectorial, se asocia a los n valores de entrada, que a su vez representan la salida de las m neuronas seleccionadas. Es reseñable que las diferentes opciones de valores de peso producen distintas funciones de red para los mismos datos de entrada, por lo que no existe idempotencia en tal sentido. La propagación hacia atrás entre las neuronas que componen la red es supervisada mediante un método de aprendizaje por el cual los valores de peso de las conexiones en la red son repetidamente ajustadas, de tal forma que se minimice la diferencia entre el vector de salidas obtenidas y el de salidas deseadas. Los algoritmos de aprendizaje basados en la propagación hacia atrás de errores tienen su principal campo de aplicación en la búsqueda de pesos óptimos para una topología de red dada, así como en sus duplas de entrada-salida.

- **Computación Evolutiva:** Paradigma computacional que toma su inspiración en la Teoría de la Evolución de Charles Darwin. Un Sistema Evolutivo Artificial es un sistema informático basado en la simulación evolutiva, en la que se comprime una población de individuos, un criterio de aptitud y los operadores genéticos que serán aplicados a la población existente para dar lugar a la siguiente. La población inicial, generada en base a un esquema aleatorio, será sujeta a mutaciones y recombinaciones, que representan principalmente la reproducción sexual entre seres vivos. A cada iteración del algoritmo, los individuos son evaluados en base a una función de aptitud, estableciendo esta una evidente analogía con la Teoría de la Evolución Darwiniana, en la que únicamente los individuos más fuertes y adaptables sobreviven. Las generaciones venideras serán obtenidas por los individuos seleccionados en base al uso de los operadores genéticos de mutación anteriormente enumerados, siendo los padres determinados por un operador de selección que refleja el principio biológico de la elección de pareja. El proceso anteriormente descrito converge, después de un relativamente elevado número de iteraciones, en una población cuasi óptima de individuos, validados por la función de aptitud. La Computación Evolutiva se

aplica, principalmente, en la creación de agentes inteligentes y la búsqueda de soluciones óptimas a un problema dado.

- **Inteligencia de Enjambre:** También conocida como Inteligencia Colectiva, la presente rama científica de la Computación Natural se define como el comportamiento dedicado a la solución de problemas que emerge de la interacción de agentes individuales naturales, tales como las hormigas, las aves o las termitas, que se comunican entre sí para actuar con su entorno. La Optimización por Enjambre de Partículas (PSO; Particle Swarm Optimization) aplica la idea anteriormente descrita al problema de la búsqueda de una solución óptima a un problema dado mediante la investigación en un espacio de soluciones, que típicamente es multidimensional. La etapa de inicialización de los algoritmos adscritos a la Inteligencia de Enjambre parte de un enjambre de partículas, en la que cada una representa una posible solución al problema. Cada partícula consta de una velocidad particular que depende de su celeridad previa, la tendencia en función a su mejor posición anterior y su tendencia con respecto al vecindario global o local óptimo. Por consiguiente, las partículas fluctúan a través de un espacio multidimensional y llegado el momento convergen en torno a un punto entre la mejor posición global y la mejor posición personal. Los algoritmos adscritos a la PSO se aplican normalmente a un amplio elenco de problemas de optimización y algoritmos de *machine learning* o aprendizaje de máquina. Por otro lado, la Inteligencia de Enjambre se aplica a modelos de hormigueros (*ant colonies*, que serán descritos exhaustivamente en posteriores epígrafes), en los que las hormigas realizan la búsqueda del mejor camino entre el nido y el alimento en base a la siguiente directriz: Las hormigas dejarán a su paso un rastro de feromonas en su vuelta al hormiguero tras haber encontrado comida, de tal forma que el resto de hormigas que busquen alimento deberán seguir dicho rastro de feromonas mientras depositan las suyas propias a su vez. De esta forma, el mejor camino será aquel que contenga un mayor número de feromonas, pues este factor desemboca inequívocamente en que ha sido transitado por un mayor número de hormigas. Tales algoritmos tienen su principal campo de aplicación en la búsqueda de caminos óptimos en problemas cuyas características se circunscriben al conocido Problema del Viajante.

- **Sistemas Inmunológicos Artificiales:** Entidades computacionales inspiradas, como su propio nombre indica, en los sistemas inmunológicos de los seres vivos. Desde el punto de vista de un sistema informático, el sistema inmunológico de los seres vivos realiza tareas complejas en un esquema paralelo y distribuido. De esta forma, los Sistemas Inmunológicos Artificiales se corresponden con abstracciones de sus homólogos naturales, haciendo énfasis en los aspectos computacionales. Sus aplicaciones incluyen la detección de virus informáticos, la detección de fallos y el reconocimiento de patrones, así como en numerosos campos relacionados con la robótica, entre otros.

- **Computación de Membranas:** La presente rama de la Computación Natural investiga los modelos informáticos obtenidos en base a la abstracción del funcionamiento de las células vivas, así como de su manera de interactuar con las membranas anexas. Un sistema de membranas genérico consiste en un conjunto de regiones, del tamaño de células, delimitados por membranas, que se sitúan en una estructura jerárquica. Cada región, comprendida en una membrana a su vez, contiene elementos y reglas de transformación que modifican a aquellos, especificando los objetos que permanecen en la región o salen de la misma. Cabe destacar que el intercambio de objetos constituye el mecanismo principal de comunicación entre regiones. La computación mediante un sistema de membranas comienza con una configuración inicial, donde la cantidad de objetos existente se establece con un valor para cada región. El proceso continúa mediante la elección, de forma paralela e indeterminista, de las reglas aplicadas a los distintos objetos. El resultado del anteriormente descrito proceso informático se recolecta desde una región predeterminada. Se destaca que el principal campo de aplicación en el ámbito natural de este tipo de algoritmos es el modelado de diferentes procesos biológicos, tales como la fotosíntesis, la inmunidad celular y la señalización entre caminos en el cuerpo de los seres vivos. Por otro lado, en cuanto al campo más puramente informático se refiere, destaca su aplicación a los gráficos por computador, la criptografía de clave pública y los algoritmos de ordenamiento.

- **Computación Amorfa:** Partiendo de la base de que se conoce como morfogénesis al desarrollo de formas bien definidas y estructuras funcionales, la Computación Amorfa se consigue mediante la interacción de células guiadas por el programa genético codificado en el ADN de un organismo vivo. Inspirada por esta idea, la Computación Amorfa pretende definir metodologías y patrones bien definidos a partir de la interacción local de un conjunto de partículas. Como paradigma de computación, el objetivo principal es el de delimitar nuevas metodologías de programación que funcionarían de forma óptima en entornos de computación amorfos, es decir, cuyos límites están difusamente delimitados. La Computación Amorfa interpreta un papel de importancia vital en la Computación Natural, pues constituye las bases de la Computación Celular, descrita en el presente epígrafe.

Finalizada la introducción al presente epígrafe, se procederá a la descripción exhaustiva del estado del arte de los algoritmos, paradigmas y metodologías involucradas en el ámbito de la presente Tesis Doctoral a continuación.

2.1.1 Fireworks Algorithm

Desde los albores de la Ciencia Computacional, los Problemas de Optimización y Búsqueda han sido uno de los principales (y recurrentes) problemas a resolver. Esta índole de problemáticas puede encontrarse en prácticamente cualquier ámbito, tanto en el espectro del ámbito académico como en el industrial.

A fin de resolver los problemas englobados en los anteriormente citados ámbitos, un amplio elenco de métodos resolutivos han sido propuestos. Recientemente, *Swarm Intelligence Optimization* se ha erigido como un tema objeto de numerosas investigaciones, especialmente orientadas a los problemas de optimización, siendo estas notablemente exitosas. Tal y como se define en [Kennedy, 2006]: “*El paradigma Swarm Intelligence alude a la habilidad resolutive de problemas que emerge de las interacciones de unidades simples de procesamiento de información*”.

Es destacable, a modo de inciso, que el término *Swarm Intelligence* (o *Inteligencia en Enjambre*, en castellano) hace referencia al comportamiento conjunto de aquellos sistemas descentralizados y auto-organizativos, independientemente de si su origen es natural o artificial [Beni, 2005]. Los sistemas de *Swarm Intelligence* consisten típicamente en una población de agentes simples interactuando localmente con sus homólogos y con su entorno. La inspiración de tales sistemas viene, frecuentemente, del espectro natural, más en concreto, de los sistemas biológicos. En ellos, los agentes siguen instrucciones extremadamente simples, y a pesar de que no existe ninguna estructura de control centralizada que dicte cómo deben comportarse los agentes individuales, las interacciones entre los mismos alcanzan una sinergia, logrando una inteligencia global mucho más compleja que la simple suma de los factores que la componen, en tanto en cuanto a los agentes individuales se refiere. Pueden encontrarse numerosos ejemplos de este escenario en sistemas de índole natural tales como *ACO*, *Bird Flocking*, *Animal Herding*, *Fish Schooling* y *Bacterial Growth*, por citar algunos, de los cuales varios se estudian en el presente capítulo.

El campo de los algoritmos de búsqueda meta-heurísticos presenta una larga historia de indagación y persecución de inspiración en los sistemas naturales [Kromer, 2014]. Empezando por algoritmos clásicos como *Genetic Algorithms* y *ACO*, las últimas décadas han resultado ser un testigo privilegiado de explosiones de fuegos artificiales o heurísticas circunscritas a un amplio espectro, como se verá en posteriores epígrafes de la presente Tesis Doctoral (desde *pájaros* hasta *abejas*, pasando por motivos fantásticos tales como los *zombies* o la *reencarnación*). Los Algoritmos Evolutivos, *PSO*, *ACO* y sus variantes dominan el campo de las metaheurísticas inspiradas en la naturaleza [Lones, 2014]. Es

también reseñable, especialmente en la línea de las metáforas anteriormente mencionadas, que un gran número de metaheurísticas similares han empezado a atraer cierto criticismo en la comunidad científica por ocultar su falta de novedad detrás de una metáfora elaborada [Campelo, 2018].

Alcanzado el presente punto, es importante reseñar que, a pesar de que los seres vivos y su carácter constituyen el aglutinante en el Paradigma de la Computación Natural, las unidades que componen el enjambre (swarm) pueden ser, a su vez, cuerpos inertes. Como se describirá pormenorizadamente a lo largo del presente epígrafe, este es, precisamente, el caso del Fireworks Algorithm (FA).

Tras el anterior inciso, es destacable que el concepto Swarm Intelligence presenta un amplio número de ramas de investigación, la gran mayoría descritos pormenorizadamente en el presente *Capítulo 2. Estado del Arte*, en el que numerosos trabajos de investigación han visto la luz recientemente. En ellos, se investigan los conceptos, comportamiento y sistemas cognitivos propios de la naturaleza, siendo tan fructíferos que han dado luz a nuevos paradigmas a su vez, como Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) y Fish School Search (FSS).

Dejando de lado el concepto más puramente semántico de Swarm Intelligence, el algoritmo que bautiza al presente epígrafe, FA, toma su inspiración en la fenomenología que acontece en la explosión de los Fuegos Artificiales:

Figura II: Explosión de Fuegos Artificiales [Zhu, 2010]

En la figura anterior se aprecia cómo el Fuego Artificial detona en un punto central, a partir del cual las chispas (sparks) nacen. (Esta aclaración conceptual es necesaria en pos de la claridad en los siguientes razonamientos.)

Ampliando el nivel de detalle en el proceso anteriormente citado, cuando un fuego artificial o firework se lanza, una lluvia de sparks llenará el espacio local alrededor del punto central. En otras palabras, es lícito definir el proceso de explosión del FA como una búsqueda en un espacio local alrededor de un punto específico donde los sucesivos fuegos artificiales se disparan a través de los sparks generados en la explosión.

La técnica Fireworks Algorithm (FA en lo sucesivo, Algoritmo de los Fuegos Artificiales), piedra angular de la presente Tesis Doctoral, interpreta un rol protagonista en el ámbito de la Computación Natural, más en concreto bajo el manto de la Swarm Intelligence. FA toma su inspiración en un proceso híbrido de optimización matemática y búsqueda, tal y como se especifica en [Boyd, 2004]. De forma paralela, FA toma su inspiración en la propia explosión de un fuego artificial [Tan, 2015], como se aprecia en la siguiente Figura:

Figura III: Símil entre la explosión de los Fuegos Artificiales y Fireworks Algorithm

La misma muestra cómo, a partir del firework central, pivotan los sparks. Dichos sparks definirán áreas de búsqueda sobre las que se intentará localizar puntos de interés. Si la búsqueda es fructífera, se lanzará de nuevo un firework en dicha zona para iterar sobre el proceso anterior hasta hallar la solución óptima. En otro caso, se continuarán lanzando fireworks en base a ciertas directrices que se describirán posteriormente en el presente epígrafe.

FA nace de la mano de los procesos de búsqueda involucrados en los problemas clásicos de la computación [Sedgewick, 1998]. Persiguiendo la finalidad de definir los FA a nivel conceptual, es pertinente clarificar ciertos conceptos: Tal y como se especifica en [Knuth, 1998], un algoritmo de búsqueda es un conjunto de

instrucciones matemáticas que persiguen la finalidad de establecer un cierto orden entre los elementos que componen una colección. Las técnicas de optimización, así como la ordenación eficiente, han sido el objetivo principal de esta suerte de algoritmos desde las primeras investigaciones en el campo [Cormen et al., 2001].

La técnica FA está especialmente concebida para la optimización multimodal de funciones matemáticas complejas, y se implementa, como ya se ha dejado entrever, siguiendo la simulación precisa del proceso de explosión de los fuegos artificiales. FA presta un interés especial a la manutención de la diversidad en los *sparks* (chispas) manteniendo dos procesos de búsqueda distintos.

En términos generales, puede afirmarse que FA constituye un esquema que pretende explorar un espacio de búsqueda mastodóntico. Dicha exploración se basa en la búsqueda de puntos de interés aleatorios confinados en una cierta medida longitudinal. Como fruto de dicha exploración, se *espera* que tales puntos de interés arrojen resultados prometedores, satisfaciendo un criterio de búsqueda preestablecido. Una vez estos puntos ofrecen resultados interesantes en términos de la función matemática a ser optimizada, una búsqueda más concentrada será lanzada en los puntos cercanos, iterando a través del algoritmo hasta encontrar una solución óptima. La siguiente Figura muestra el Diagrama de Flujo de un FA [Tan et al., 2010]:

Figura IV: Diagrama de Flujo de un FA

Como puede apreciarse en la figura anterior, y en lo referido al esquema operacional de FA, el algoritmo comienza con la selección de un número predefinido, N , de posiciones iniciales. N fireworks (fuegos artificiales) serán lanzados en estas ubicaciones, a su vez que, iterativamente, los fireworks lanzarán sparks (chispas). La localización de dichos sparks será analizada para evaluar su calidad, finalizando si se considera que una solución óptima se ha encontrado. Debe advertirse que, en el contexto de un problema de optimización, no suele existir una certeza total de haber encontrado un punto óptimo absoluto, por lo que debe el algoritmo finalizará si se considera que el punto encontrado puede considerarse óptimo. Por ejemplo, es frecuente considerar un punto como óptimo siempre y cuando se hayan efectuado N iteraciones posteriores sin haber encontrado un óptimo mejor, donde N sea un número perteneciente al conjunto natural lo suficientemente elevado dependiendo del escenario. En caso de no encontrar un punto considerado óptimo, FA lanzará de nuevo N fireworks en N posiciones, iterando una vez tras otra sobre el anteriormente citado proceso hasta que la solución óptima se encuentre. En otras palabras, los N fireworks lanzarán a su vez N sparks. El área de estas sparks, será el espacio de búsqueda. En caso de que alguna de ellas arroje resultados de interés, se lanzará en cada área N fireworks de nuevo, hasta que se halle la solución óptima.

Habiendo introducido y definido FA, es pertinente orientar el foco hacia las características principales que definen la naturaleza más puramente atómica del algoritmo. Tal y como ocurre en numerosos paradigmas adscritos al ámbito de la Computación Natural y la Swarm Intelligence, FA no será una excepción a la existencia de los denominados *operadores* [Tan, 2015], que definirán su estructura y comportamiento. A continuación se presentan los anteriormente citados *operadores* de FA:

- *Explosion Operator*: En lo referido a la inicialización de FA, se generan N fireworks de forma aleatoria. La explosión de los mismos será un proceso clave en FA, interpretando un papel fundamental en el devenir del algoritmo. Dentro del ámbito del operador explosivo, existen tres principales factores, relatados seguidamente:
 - *Explosion Strength*: Simula su homólogo en el mundo real. Cuando un firework explota, este se desvanece en un segundo, para dar paso a pequeños fogonazos a su alrededor. FA determina, en primer lugar, el número de sparks o chispas, para después calcular la amplitud de cada explosión. A través de la observación de las curvas en algunas funciones de optimización típicas, puede apreciarse que hay más puntos con un aceptable valor de aptitud alrededor del punto óptimo que en aquellas áreas más alejadas del mismo. Por consiguiente, los fireworks con mejores valores de aptitud o *fitness* producen más sparks. En términos matemáticos, la fuerza de la explosión puede determinarse con la siguiente fórmula, donde S_i es el número de sparks para cada individuo, m es un valor constante en función del número total de sparks e $Y_{\max}$ simboliza el valor fitness del peor individuo entre los N que componen la población:

$$S_i = m * \frac{Y_{\max} - f(x_i) + \varepsilon}{\sum_{i=1}^N (Y_{\max} - f(x_i)) + \varepsilon}$$

De esta forma, se evita merodear cerca del punto óptimo sin localizarlo. Para los fireworks con peores valores fitness, el número de sparks generadas será menor, evitando así una carga computacional innecesaria o de ínfimo valor neto. No obstante, estos fireworks son útiles para explorar el espacio de búsqueda, evitando la convergencia prematura de FA. La siguiente figura muestra una explosión óptima en (a) y una homóloga de menor

calidad en (b). Se aprecia, a su vez, que en (a) el firework produce más sparks con una amplitud menor, mientras que en (b) ocurre exactamente lo opuesto; menos sparks con más área de análisis.

Figura V: Explosión óptima de un firework en (a) y homóloga de baja calidad en (b)

- *Explosion Amplitude*: El estudio y la observación de las curvas en numerosas funciones de optimización típicas [Tan, 2016] muestra cómo los puntos alrededor del punto local óptimo, así como del global, siempre tienen mejores valores de aptitud o *fitness*. Por ello, el control de la amplitud de la explosión es clave en el devenir del FA; la amplitud de los fireworks con mejor valor de aptitud será gradualmente reducida, ayudando al algoritmo a encontrar los puntos óptimos locales y el global. En contraposición, los fireworks con peores valores fitness explorarán el área de búsqueda a través de una gran amplitud explosiva. Manteniendo la nomenclatura introducida en la anterior fórmula, la amplitud de la explosión viene dada por la siguiente fórmula, donde A' es la constante determinada por el sumatorio de amplitudes, A_i denota la amplitud individual, A' es la constante dada por la suma de todas las amplitudes, $Y_{\min}$ el valor de aptitud para el mejor individuo entre los N disponibles, $f(x_i)$ la aptitud del individuo x_i y ϵ una constante “centinela” para evitar que el denominador sea 0:

$$A_i = \hat{A} * \frac{f(x_i) - Y_{\min} + \varepsilon}{\sum_{i=1}^N (f(x_i) - Y_{\min}) + \varepsilon}$$

- *Displacement operator*: Una vez se calcula la amplitud de la explosión, es fundamental calcular el desplazamiento entre dicha amplitud. FA se comporta de manera simple en este caso, utilizando un desplazamiento aleatorio. Así, cada firework tendrá su número de explosiones específico y la amplitud de sus sparks. FA genera valores de desplazamiento aleatorios para cada amplitud, asegurando la diversidad genotípica o de población. A través del operador explosivo, cada firework generará ráfagas de sparks, que ayudarán a hallar el óptimo global de una función de optimización. El pseudo-algoritmo asociado a la generación de sparks figura seguidamente:

```

1: Initialization, calculate the fitness value  $f(x_i)$  for each firework.
2: Calculate the number of sparks  $S_i$ .
3: Calculate the amplitude of sparks  $A_i$ .
4:  $z = rand(1, dimension)$  //randomly choose  $z$  dimensions
5: for  $k = 1 \rightarrow dimension$  do
6:   if  $k \in z$  then
7:      $x_i^k = x_i^k + U(-A_i, A_i)$ 
8:   end if
9: end for

```

Figura VI: Algoritmo de generación de sparks en FA

- *Gaussian Mutation Operator*: A fin de mejorar la diversidad de la población, se presenta la mutación Gaussiana en el ámbito del FA. Este operador se encarga de producir sparks mediante la selección de un firework perteneciente a la población, para luego aplicar una mutación Gaussiana en dimensiones aleatoriamente seleccionadas. Para esta mutación, los nuevos sparks se generan entre el mejor firework y los homólogos preseleccionados, tal y como se muestra en la siguiente figura:

Figura VII: Mutación Gaussiana en el espectro del FA

Como puede apreciarse, cuando un spark yaza más allá del límite superior o inferior, la *mapping rule*, que será definida a continuación, será encargada de realizar el proceso de mapeado del spark a una nueva localización en el espacio de búsqueda. La Mutación Gaussiana se calcula mediante la aplicación de la siguiente entidad algorítmica:

- 1: Calculate the fitness value $f(x_i)$ for each firework.
- 2: Calculate the coefficient $g = \mathcal{N}(1, 1)$.
- 3: $z = rand(1, dimension)$ //randomly select z dimensions
- 4: **for** $k = 1 \rightarrow dimension$ **do**
- 5: **if** $k \in z$ **then**
- 6: $x_i^k = x_i^k * g$
- 7: **end if**
- 8: **end for**

Figura VIII: Algoritmo de Mutación Gaussiana en FA

- *Mapping Rule*: Siempre y cuando un firework se encuentre cerca del límite del espacio de búsqueda, y mientras su amplitud de la explosión cubra tanto el espacio de búsqueda permitido como el que escape a dicho ámbito, podría darse el caso de que existan sparks fuera de los límites del espacio. Por tanto, estas sparks serían inútiles, ya que sería imposible que arrojaran datos de interés. Así, necesita recuperarse para introducirla de nuevo en el espacio de búsqueda aceptado. La *Mapping Rule* o Regla de Mapeo es, precisamente, el operador encargado de lidiar con esta casuística, asegurando que todas y cada una de las sparks se encuentran en los límites aceptados, trayendo sparks de vuelta al espacio de

búsqueda si se generara fuera del mismo. El cálculo del *Mapping Rule*, donde x_i^k representa la posición de cualquier spark fuera de rango, $X_{UB,k}$ y $X_{LB,k}$ el máximo y el mínimo límite de la posición de sparks y % representa la operación modular, figura a continuación:

$$x_i^k = X_{LB,k} + x_i^k \% (X_{UB,k} - X_{LB,k})$$

- *Selection Strategy*: Una vez se aplique el operador explosivo, de mutación y la mapping rule, alguna de las sparks generadas necesitarán seleccionarse y propagarse hacia las siguientes generaciones. De esta manera, entra en juego el operador *Selection Strategy*, que pretende seleccionar qué sparks se propagarán hacia las siguientes iteraciones del algoritmo. En dicha selección, la mejor spark siempre se guardará para futuras generaciones, mientras que el resto serán seleccionadas basadas en la distancia necesaria para mantener una diversidad aceptable en la población. El individuo que esté más lejos de sus homólogos tiene una mayor probabilidad de ser seleccionado en comparación con aquellos cercanos a otros. La fórmula acorde a *Selection Strategy*, donde $d(x_i, x_j)$ denota la distancia Euclídea entre dos individuos, es la siguiente:

$$R(x_i) = \sum_{j=1}^K d(x_i, x_j) = \sum_{j=1}^K \|x_i - x_j\|$$

Además de los operadores anteriormente descritos, existen otras características de FA que merece la pena destacar. Entre ellas, destaca el factor de instantaneidad del algoritmo. En cada iteración se calcula el número de sparks y la amplitud de la explosión, dependiendo del valor de fitness de los fireworks, como se especificó anteriormente. Finalmente, la mejor spark es seleccionada como la primera a transmitir a las nuevas iteraciones, mientras que sobre el resto se aplica la *Selection Strategy*, ya descrita. Las N sparks seleccionadas serán tratadas como fireworks en la siguiente generación, de manera que se descarta el resto. Durante este proceso, destaca que ningún spark o firework es almacenado para posterior computación, destacando así la instantaneidad de FA.

A su vez, como en otros algoritmos pertenecientes al espectro de la *Swarm Intelligence*, cada firework presente en el algoritmo carece, totalmente, de cualquier tipo de información sobre el resto de sus homólogos. Además de su información propia, es suficiente con conocer someramente su entorno más puramente directo [Xiao, 2007], llevando a cabo simples directrices para cumplir su cometido. Por esta razón, puede inferirse que la efectividad de FA no proviene

de complejos sistemas, sino, como en numerosos otros ámbitos de la ingeniería, en la suma pequeña, pero crucial, de valores infinitesimales. Dado que FA se compone de agentes simples, es lícito afirmar que se caracteriza también por su simpleza.

En cuanto a la interacción entre fireworks se refiere, destaca que esta se compone de una suerte de híbrido entre la competición y la colaboración. Cada firework pretende que su búsqueda sea la más efectiva, a su vez sin torpedear el proceso de otro firework. Esto resulta en un sistema en el que la simbiosis da lugar a una alta inteligencia colectiva; mientras que cada firework, siguiendo el razonamiento del párrafo anterior, es una entidad tan simple como atómica, el algoritmo en sí es capaz de resolver problemas matemáticos de optimización y de búsqueda altamente complejos [Liu et al., 2004].

Es lícito destacar que el FA, por otro lado, está impregnado de un paralelismo en su distribución, que lo hace especialmente indicado para la programación basada en multi-threading; en cada iteración, cada firework explota y busca en un espacio distinto, para que, finalmente, las sparks y los fireworks se combinen en pos de la elección de un subconjunto a propagar a la siguiente generación. En cada una de ellas, FA es capaz de ser paralelizado. No ocurre lo mismo, desafortunadamente, con la población, de carácter más centralizado, en el que la diversidad es crucial para el rendimiento de FA. Si se respeta la diversidad de la población, el algoritmo podrá variar entre óptimos locales, lo que provocará la convergencia hacia el óptimo global. Por tanto, el valor óptimo será más fácilmente identificable si la población es fuertemente dispar, para que la convergencia no sea afectada de forma significativa.

Siguiendo en la línea de características de FA, y pese a los múltiples trabajos de investigación que han visto recientemente la luz sobre la materia, el número de sparks involucradas en el algoritmo es, a día de hoy, incierto o posible de determinar basándose en la complejidad de un problema dado. El número de sparks y fireworks puede crecer o decrecer, puesto que la variación en los individuos de la población podría efectivamente solucionar el problema dado. Este importante factor permite afirmar que FA es tan extensible como adaptable, puesto que no es necesario definir un problema en la forma de una expresión matemática totalmente cerrada para su resolución. FA, por tanto, es un algoritmo de corte adaptativo [Tan, 2008].

Es, por otro lado, lícito preguntarse por la comparación entre FA y otros esquemas de optimización matemática/búsqueda análogos. En tal sentido, [Tan, 2015] presenta un fructífero estudio en el que, a fin de validar las curvas de convergencia de FA, las técnicas algorítmicas Clonal Particle Swarm Optimization (CPSO) y Standard Particle Swarm Optimization (SPSO) son comparadas. A

continuación se presenta la figura que contiene el modelo en el que SPSO se apoya:

Figura IX: Modelo de Standard Particle Swarm Optimization: Actualización posicional a través de una topología estrella ($gBest$) y un término inercial, w .

Persiguiendo la validación de la convergencia en FA, el estudio presenta un conjunto de 8 diferentes funciones de Benchmarking (Sphere, Rosenbrock, Ellipse, Cigar, Rastrigin, Griewank, Tablet y Schwefel), medidas sobre 20 ejecuciones independientes en los algoritmos. El estudio alcanza la conclusión de que FA presenta una velocidad superior a sus competidores, CPSO y SPSO. Esta afirmación es empíricamente demostrada por la Figura que se presenta seguidamente:

Figura X: Resultados sobre las Funciones de Benchmarking fruto de la comparación entre FA, CPSO y SPSO. [Ying, 2015]

La información involucrada en la anterior Figura, sumada a los resultados arrojados por la comparación con la Función de Benchmark Ackley, pueden analizarse en la siguiente Figura, donde la Media Estadística y la Desviación Típica para FA son con creces más ágiles en contraposición a sus competidores, CPSO y SPSO:

Function	FA's mean (StD)	CPSO's mean (StD)	SPSO's mean (StD)
Sphere	0.000000 (0.000000)	11857.425781 (3305.973067)	24919.099609 (3383.241523)
Rosenbrock	19.38330 (11.94373)	2750997504.000000 (1741747548.420642)	5571942400.000000 (960421617.568024)
Rastrigrin	0.000000 (0.000000)	10940.148438 (3663.484331)	24013.001953 (4246.961530)
Griewank	0.000000 (0.000000)	3.457273 (0.911027)	7.125976 (0.965788)
Ellipse	0.000000 (0.000000)	2493945.500000 (1199024.648305)	5305106.500000 (1117954.409340)
Cigar	0.000000 (0.000000)	122527168.000000 (28596381.089661)	149600864.000000 (13093322.778560)
Tablet	0.000000 (0.000000)	15595.107422 (8086.792234)	42547.488281 (8232.221882)
Schwefel	4.353733 (1.479332)	8775860.000000 (1217609.288290)	6743699.000000 (597770.084232)
Ackley	0.000000 (0.000000)	15.907665 (1.196082)	18.423347 (0.503372)

Figura XI: Media Estadística y Desviación Típica de las soluciones encontradas por FA, CPSO y SPSO en nueve Funciones de Benchmarking sobre 20 ejecuciones independientes de 10000 evaluaciones de función. [Ying, 2015]

A modo de conclusión, se destaca que FA encuentra soluciones brillantes con evaluar las Funciones de Evaluación sólo un millar de veces, reflejando así la rápida velocidad de convergencia de FA cuando se compara con su homóloga de CPSO o SPSO.

Por otro lado, en cuanto a las aplicaciones de FA, es destacable [Ying, 2016]. En dicho estudio, FA es aplicado a la creación de un sistema de minería de datos para aplicaciones de *low-ranking*, tales como la búsqueda web. Típicamente, estos sistemas ven reducidos su capacidad de almacenamiento y requisitos en tiempo de ejecución, así como la información (con ruido presente) en lo referido a asociaciones esenciales en el ámbito de estos sistemas. El paradigma Non-negative Matrix Factorization (NMF), cuyos parámetros se describen pormenorizadamente en [Lee et al., 1999], alcanza una aproximación *low-ranking* que verifica que las constantes de no-negatividad en NMF se satisfacen. NMF aproxima una matriz de datos por ciertos factores que dependen de la naturaleza del algoritmo, siendo necesario que los datos contenidos en la matriz anteriormente citada sean no vacíos y mayores que cero.

Otra interesante aplicación de FA puede apreciarse en [Xue et al., 2016], donde una versión avanzada del FA se aplica al ajuste de parámetros de Derivada-Integral-Proporcional (PID; Proportional-Integral-Derivative).

Es destacable que un PID se corresponde con un mecanismo de retroalimentación en bucle de control, ampliamente aplicado en sistemas de control industrial [Bennett, 2016], así como en un ancho espectro de aplicaciones que requieren un control continuamente modulado. Un PID calcula, de manera sostenida, un valor de error como la diferencia entre un setpoint (SP) y una variable de proceso medida (PV). De esta manera, se aplica una corrección basada en términos proporcionales, integrales y derivativos, bautizando de esta manera al PID, que opera bajo los diseños del siguiente Diagrama de Bloque:

Figura XII: Diagrama de Bloque de un controlador Proporcional, Integral y Derivativo (PID)
[Bellman, 2015]

Con respecto a la fórmula de operación del PID, es reseñable que existen numerosas variaciones, siendo la más popular en la industria la denominada estándar. En la misma, la ganancia (K_p) se aplica a los términos I_{out} y D_{out} , arrojando la siguiente expresión, en la que T_i es el tiempo integral y T_d el derivativo:

$$MV(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{d}{dt} e(t) \right)$$

Es destacable que esta fórmula estándar es la utilizada en lo referido al diseño digital de controladores PID en microcontroladores o dispositivos FPGA (Field Programmable Gate Array), lo que requiere un proceso de discretización [Scribd, 2004] en la misma por el método de diferencias finitas:

$$\int_0^{t_k} e(\tau) d\tau = \sum_{i=1}^k e(t_i) \Delta t$$

Donde el término es aproximado como el cociente entre dos valores, donde el incremento de t actúa como denominador:

$$\frac{de(t_k)}{dt} = \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

Por consiguiente, obteniendo la fórmula final del algoritmo de implementación de la discretización del controlador PID en su versión para el microcontrolador, para la que son necesarias las definiciones numéricas de la primera y segunda derivada, la solución de $u(t_k)$ y la obtención final de:

$$u(t_k) = u(t_{k-1}) + K_p \left[\left(1 + \frac{\Delta t}{T_i} + \frac{T_d}{\Delta t} \right) e(t_k) + \left(-1 - \frac{2T_d}{\Delta t} \right) e(t_{k-1}) + \frac{T_d}{\Delta t} e(t_{k-2}) \right]$$

$$T_i = K_p / K_i, T_d = K_d / K_p$$

Introducido el concepto de PID, [Xue et al., 2016] propone una metodología de diseño de controladores PID de alta calidad, aplicando una variación del FA basada en el principio de auto-adaptación. Para ello, propone una Gaussiana bimodal especialmente concebida para la optimización del controlador PID mediante parametrización. Las simulaciones de esta propuesta arrojan que la variación de FA es efectiva y fácil de implementar en pos de la resolución de problemas asociados al control del PID, así como diferentes funciones de transferencia.

Es importante destacar ciertas variaciones en dicho algoritmo que ponen de evidente manifiesto la gran versatilidad de FA, pues, en el mismo, se modifican principalmente tres grandes factores: El operador de explosión que domina el FA, su mutación (basándose en una Función Gaussiana Bimodal) y la selección de las ubicaciones de la explosión.

Así, el número de sparks generados por la explosión, que puede ser vista como un proceso de búsqueda alrededor de un punto específico, se respalda, en el caso del estudio bajo análisis, por la fórmula (similar a la anteriormente presentada):

$$N_i = \text{round} \left(N_c \cdot \frac{f_{\max} - f(X_i) + \varepsilon}{\sum_{i=1}^N (y_{\max} - f(X_i)) + \varepsilon} \right)$$

Por otro lado, de forma análoga, la amplitud de la explosión en la variación algorítmica de FA, vendrá dada en esta ocasión por:

$$A_i = A_c \cdot \frac{f(X_i) - f_{\min} + \varepsilon}{\sum_{i=1}^N (f(X_i) - f_{\min}) + \varepsilon}$$

La mutación en la explosión basada en la Función Gaussiana Bimodal se determinará, a su vez, por la expresión:

$$r = \frac{Xf_i^z - X_{\min}^z}{X_{\max}^z - X_{\min}^z}$$

Mientras que, finalmente, la selección de las localizaciones de explosión se refleja en las fórmulas:

$$p(i) = \frac{R(i)}{\sum_{j \in K} R(j)}$$

$$R(i) = \sum_{j \in K} d(i, j) = \sum_{j \in K} \|X_i - X_j\|$$

Se referencia a [Xue et al., 2016] para un mayor nivel de detalle matemático en el elenco de fórmulas que componen la modificación del algoritmo FA para el control PID en microcontroladores, que es simulado frente a funciones de benchmarking clásicas para medir su rendimiento.

Tras haber descrito FA en profundidad en base a un enfoque *bottom-up*, se presenta a continuación el pseudocódigo adscrito a tal entidad algorítmica:

```

1: Randomly select  $N$  locations for fireworks
2: while terminal condition is not met do
3:   Set off  $N$  fireworks, respectively, at the  $N$  locations:
4:   for all fireworks  $x_i$  do
5:     Calculate the number of sparks as  $S_i$ 
6:     Calculate the amplitude of sparks as  $A_i$ 
7:   end for
8:   //  $\hat{m}$  is the number of sparks generated by Gaussian mutation
9:   for  $k = 1 \rightarrow \hat{m}$  do
10:    Randomly select a firework  $x_i$  and generate a spark
11:   end for
12:   select the best spark and the other sparks according to selection strategy
13: end while

```

Una vez descrita la técnica FA, y partiendo del pseudocódigo introducido anteriormente, se procede a la presentación de la implementación del mismo, fielmente realizada en el lenguaje de programación Java (Version 1.8). Se aprovecha para informar al lector de que el código fuente de la siguiente aplicación, al igual que el resto de software involucrado en la presente Tesis Doctoral, puede encontrarse en el repositorio GitHub dedicado; <https://github.com/ClemencioMoralesLucas/PhD>. Más en concreto, el código perteneciente a FA se ubica bajo la ruta <https://github.com/ClemencioMoralesLucas/PhD/tree/master/src/phd/cml/fireworks>. Dado que dicho software (y todo aquel fruto de esta Tesis Doctoral) ha sido implementado bajo las directivas del Paradigma *Test Driven Development* (TDD), sus tests unitarios se encuentran en <https://github.com/ClemencioMoralesLucas/PhD/tree/master/test/phd/cml/firework> [S](#).

En lo referido a la implementación de FA, es recomendable optar por un enfoque *bottom-up*. Es reseñable que este enfoque demuestra ser superior a su rival, *top-down*, especialmente en el espectro de la Programación Orientada a Objetos, puesto que permite reflejar sistemas complejos de manera más fiel [Stewart et al., 2015] y con una menor presencia de lagunas lógicas.

A fin de estudiar la técnica FA en su versión clásica, se implementará un framework de *benchmarking*, que contendrá las funciones matemáticas más recomendadas y típicas [Thomas, 1995] para llevar a cabo este tipo de mediciones. El nombre de cada una de estas entidades puede encontrarse en el enum `BenchmarkFunctions.java`:

```

package phd.cml.fireworks;

/**
 * Created by Clemencio Morales Lucas.
 */

public enum BenchmarkFunctions {

    SPHERE, SCHWEFEL, ROSENBROCK, ACKLEY, GRIEWANKS, RASTRIGIN, PENALIZED,
    SIX_HUMP_CAMEL_BACK, GOLDSTEIN, SCHAFFER, AXIS_PARALLEL,
    ROTATED_HYPER_ELLIPSOID;
}

```

Por tanto, FA será ejecutado para resolver las anteriormente presentadas funciones de benchmarking, cuya implementación estará contenida en la clase `BenchmarkFunction.java`. Dado que este fichero es especialmente extenso, se presenta, a modo de ejemplo, la implementación de una de las funciones de benchmarking matemático anteriormente expuestas; *Penalized Benchmark Function*:

```

/**
 * Penalized Function Bounds[-50,50] dim30 optimum=1D ini[25,50]
 * @param inputValues
 * @return result of the computation
 */
private double penalizedBenchmarkFunction(final double[] inputValues) {
    double aux = 0, result = 0;
    int i;
    for (i = 0; i < inputValues.length; i++) {
        aux = aux + miu_benchmark(inputValues[i], MIU_BOUND, MIU_MULTIPLIER_FACTOR,
MIU_POWER);
    }
    for (i = 0; i < inputValues.length - 1; i++) {
        result = result + Math.pow(inputValues[i] - 1, SQUARE_VALUE) * (IDENTITY_VALUE +
        Math.pow(Math.sin(PENALIZED_BENCHMARK_FUNCTION_SIN_VALUE * Math.PI *
        inputValues[i + 1]), SQUARE_VALUE));
    }
    result = result + Math.pow(Math.sin(PENALIZED_BENCHMARK_FUNCTION_SIN_VALUE * Math.PI *
    inputValues[0]), DOUBLE_SQUARE_VALUE);
    result = result + Math.pow(inputValues[inputValues.length - 1] - IDENTITY_VALUE,
    SQUARE_VALUE) * (IDENTITY_VALUE + Math.pow(Math.sin(SQUARE_VALUE * Math.PI *
    inputValues[inputValues.length - 1]), SQUARE_VALUE));
    result = result * PENALIZED_BENCHMARK_FUNCTION_MULTIPLIER;
    return result + aux;
}

```

Es digno de mención que, persiguiendo la máxima claridad y legibilidad en el código fuente, características deseables para cualquier software, todas las constantes están almacenadas en el interface Java `BenchmarkFunctionConstants`. El mismo diseño ha sido aplicado en el caso de `FireworkAlgorithmConstants`.

Recapitulando, a tenor de la metodología bottom-up, y como el atento lector puede intuir, el primer paso a llevar a cabo es detectar la entidad más atómica del FA: el *spark*. De forma esencial, esta entidad consta de una posición en el espacio de búsqueda, una necesidad de ser evaluado para comprobar si encontró un punto de interés y un valor que lo cuantifique. Por ende, la siguiente clase refleja tal objeto a nivel lógico. Es reseñable que, persiguiendo la presentación de únicamente aquella información especialmente útil del software, se obviarán los métodos de utilidad y/o estándares, tales como `toString()`, `equals()`, `hashCode()`, `get()` y `set()`, siempre y cuando sea posible:

```
package phd.cml.fireworks;

import java.util.Arrays;

/**
 * Created by Clemencio Morales Lucas.
 */

public class Spark {

    private double [] position;
    private boolean evaluated;
    private double value;

    public Spark() {
        this.setEvaluated(false);
    }

    public void setPosition(final double [] receivedPosition) {
        this.position = new double [receivedPosition.length];
        for (int i = 0; i < receivedPosition.length; i++) {
            this.setPositionAtIndex(i, receivedPosition[i]);
        }
        this.setEvaluated(false);
    }

    public double getValue(final BenchmarkFunction function) {
        if(!evaluated) {
            this.setEvaluated(true);
            value = function.getBenchmarkingFunctionValue(position);
            return value;
        } else {
            return value;
        }
    }

    @Override
    public int hashCode() {
        int result;
        long temp;
        result = Arrays.hashCode(position);
        result = 31 * result + (evaluated ? 1 : 0);
        temp = Double.doubleToLongBits(value);
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        return result;
    }
}
```

Como puede apreciarse en la clase parcialmente expuesta (siéntase libre el lector de acudir al repositorio GitHub anteriormente expuesto para estudiar el código en su completitud), un spark consta de una posición, un *flag* de evaluación y un valor. A su vez, y como indica la lógica, dos sparks serán iguales siempre y cuando se sitúen en la misma posición, tengan el mismo valor y estén igualmente evaluados.

Una vez definido el *spark* a nivel de entidad software, se está en posición de abordar el FA a tenor del pseudocódigo anteriormente mostrado en este epígrafe. En primer lugar, el FA debe conocer el entorno del problema en particular, los sparks involucrados, límites (inferiores y superiores), parámetros de la función Gaussiana, dimensiones disponibles y, en este caso particular, la función de *benchmarking* a evaluar. Estas premisas, junto con detalles de implementación adicionales y auto-explicativos, figuran en el siguiente constructor, junto con sus pertinentes atributos:

```
public class FireworkAlgorithm implements FireworkAlgorithmConstants {

    private Spark[] fireworks;
    private Spark[][] sparks;
    private Spark[] gaussianSparks;
    private int locationsNumber;
    private int maximumSparksNumber;
    private double lowBound;
    private double highBound;
    private double maximumAmplitudeNumber;
    private int gaussianSparksNumber;
    private double[] maximumBound;
    private double[] minimalBound;
    private int dimension;
    private String infoFilePath;
    private int numGenerations;
    private double optimumValue;
    private int numFunctionEvaluations;
    BenchmarkFunction benchmarkFunction;

    public FireworkAlgorithm(final int locationsNumber, final int maximumSparksNumber, final double
lowBound,
                             final double highBound, final double maximumAmplitudeNumber,
                             final int gaussianSparksNumber,
                             final double[] maximumBound, final double[] minimalBound, final String infoFilePath,
                             final BenchmarkFunction benchmarkFunction) {
        this.locationsNumber = locationsNumber;
        this.maximumSparksNumber = maximumSparksNumber;
        this.lowBound = lowBound;
        this.highBound = highBound;
        this.maximumAmplitudeNumber = maximumAmplitudeNumber;
        this.gaussianSparksNumber = gaussianSparksNumber;
        this.maximumBound = maximumBound;
        this.minimalBound = minimalBound;
        this.dimension = this.maximumBound.length;
        this.infoFilePath = infoFilePath;
    }
}
```

```

    this.benchmarkFunction = benchmarkFunction;
}

```

Como puede apreciarse en el anterior código, el FA será, en el mismo instante de instanciación, consciente de la casuística que le rodea, tanto a él mismo como al problema a resolver. Para ello, se presenta el método `launch()`, codificado a imagen y semejanza del pseudo algoritmo del FA mostrado en este mismo epígrafe:

```

public double launch() {
    selectNInitialLocations();
    while (!stopCriteria()) {
        setOffNFireworks();
        selectNLocations();
    }
    return optimumValue;
}

```

En el método puede apreciarse que, tras seleccionar N ubicaciones, y mientras no se hayan cumplido las condiciones de parada, se dispararán N fireworks para seleccionar N ubicaciones de nuevo, comenzando las fases iterativas del algoritmo. (Estos métodos se irán presentando pertinentemente.) Comenzando con la selección de las N ubicaciones, se introduce dicho método, atómicamente de inicialización como puede intuirse:

```

private void selectNInitialLocations() {
    numGenerations = 0;
    numFunctionEvaluations = 0;
    fireworks = new Spark[locationsNumber];
    double[] randomPosition = new double[dimension];

    for (int i = 0; i < locationsNumber; i++) {
        fireworks[i] = new Spark();
        for (int j = 0; j < dimension; j++) {
            randomPosition[j] = maximumBound[j] - Math.random() * (maximumBound[j] - minimalBound[j]) *
QUARTER;
        }
        fireworks[i].setPosition(randomPosition);
    }
}

```

Por ende, se instancian las ubicaciones iniciales en el ámbito del espacio de búsqueda. Posteriormente, se deben lanzar los N fireworks necesarios:

```

private void setOffNFireworks() {
    numGenerations++;
    double maximumValue = fireworks[0].getValue(benchmarkFunction);
    double minimumValue = fireworks[0].getValue(benchmarkFunction);
    int i;
    for (i = 1; i < locationsNumber; i++) {
        if (fireworks[i].getValue(benchmarkFunction) > maximumValue) {
            maximumValue = fireworks[i].getValue(benchmarkFunction);

```



```

    }
    if (fireworks[i].getValue(benchmarkFunction) < minimumValue) {
        minimumValue = fireworks[i].getValue(benchmarkFunction);
    }
}
double sumMaxDiff = 0.0;
double sumMinDiff = 0.0;

for (i = 0; i < locationsNumber; i++) {
    sumMaxDiff += maximumValue - fireworks[i].getValue(benchmarkFunction);
    sumMinDiff += fireworks[i].getValue(benchmarkFunction) - minimumValue;
}

int[] sparksNumber = new int[locationsNumber];
getNumberOfSparksForAllFireworks(maximumValue, sumMaxDiff, sparksNumber);

double explosionAmplitude[] = new double[locationsNumber];
getExplosionAmplitudeForAllFireworks(minimumValue, sumMinDiff, explosionAmplitude);

sparks = new Spark[locationsNumber][];
double[] temporalPosition = generateSparksForAllFireworks(sparksNumber, explosionAmplitude);

gaussianSparks = new Spark[gaussianSparksNumber];
gaussianExplode(temporalPosition);
}

```

Durante el todo el desarrollo del presente código fuente se ha velado por impregnarlo de un carácter autoexplicativo, razón por la que no se describirá en exceso su comportamiento a fin de evitar redundancias. No obstante, baste decir que, tras establecer los límites de la función de benchmarking actual, los fireworks son lanzados, generando a su vez una lluvia de sparks, sobre la que se aplican los operadores expuestos en el epígrafe, como la amplitud de la explosión. Adviértase a su vez que esta explosión se encuentra íntimamente ligada con la función Gaussiana:

```

private void gaussianExplode(double[] temporalPosition) {
    int i;
    double[] fireworkPosition;
    int k, j;
    Random rand;
    for (k = 0; k < gaussianSparksNumber; k++) {
        gaussianSparks[k] = new Spark();
        rand = new Random();
        i = Math.abs(rand.nextInt()) % locationsNumber;
        fireworkPosition = fireworks[i].getPosition();
        boolean[] randFlag = new boolean[dimension];
        Arrays.fill(randFlag, false);

        int numExplosionDirections = (int) (dimension * Math.random());
        int randomCount = 0;
        int temporaryRandom;
        while (randomCount < numExplosionDirections) {
            temporaryRandom = Math.abs(rand.nextInt()) % dimension;
            if (!randFlag[temporaryRandom]) {
                randFlag[temporaryRandom] = true;
            }
        }
    }
}

```

```

        randomCount++;
    }
}

double gaussianCoefficient = GAUSSIAN_COEFFICIENT_BASE + rand.nextGaussian();
for (j = 0; j < dimension; j++) {
    if (randFlag[j]) {
        temporalPosition[j] = fireworkPosition[j] * gaussianCoefficient;
        if (temporalPosition[j] < minimalBound[j] || temporalPosition[j] > maximumBound[j]) {
            double abspos = Math.abs(temporalPosition[j]);
            while (abspos >= 0) {
                abspos -= (maximumBound[j] - minimalBound[j]);
            }
            abspos += (maximumBound[j] - minimalBound[j]);
            temporalPosition[j] = minimalBound[j] + abspos;
        }
    } else {
        temporalPosition[j] = fireworkPosition[j];
    }
}
gaussianSparks[k].setPosition(temporalPosition);
}
}

```

Llegado este punto se presenta el método encargado de la explosión algorítmica *per se*, donde la posición del firework será pertinente actualizada, así como el método de selección de las N ubicaciones pertinentes y el cálculo de probabilidades para cada una de ellas:

```

private void explode(double v, double[] temporalPosition, double[] fireworkPosition, boolean[] randomFlag)
{
    int j;
    double displacementRatio = v * (Math.random() - HALF) * DOUBLE_COEFFICIENT;
    for (j = 0; j < dimension; j++) {
        if (randomFlag[j]) {
            temporalPosition[j] = fireworkPosition[j] + displacementRatio;
            if (temporalPosition[j] < minimalBound[j] || temporalPosition[j] > maximumBound[j]) {
                double absolutePosition = Math.abs(temporalPosition[j]);
                while (absolutePosition >= 0) {
                    absolutePosition -= (maximumBound[j] - minimalBound[j]);
                }
                absolutePosition += (maximumBound[j] - minimalBound[j]);
                temporalPosition[j] = minimalBound[j] + absolutePosition;
            }
        } else {
            temporalPosition[j] = fireworkPosition[j];
        }
    }
}

private void selectNLocations() {
    Spark bestSpark = selectBestLocation();
    outputBestValue(bestSpark);
    final int fireworkSparksNumber = calculateNumberOfFireworksAndSparks();
    numFunctionEvaluations += fireworkSparksNumber;
    final double[][] fireworkAndSparksPositions = storeFireworksAndSparks(fireworkSparksNumber);
}

```

```

    final double[] cumulativeProbability = calculateProbabilityForEachLocation(fireworkSparksNumber,
fireworkAndSparksPositions);
    final int[] nextLocations = selectLocationsDueToProbability(fireworkSparksNumber, cumulativeProbability);
    setNextGenerations(bestSpark, nextLocations);
}

```

```

private double[] calculateProbabilityForEachLocation(final int fireworkSparksNumber, final double[][]
fireworkAndSparksPositions) {
    int i, j, k;
    double[] probabilityOfSelection = new double[fireworkSparksNumber];
    double sumOfProbability = 0, auxDistance;
    for (i = 0; i < fireworkSparksNumber; i++) {
        probabilityOfSelection[i] = 0;
        for (j = 0; j < fireworkSparksNumber; j++) {
            auxDistance = 0;
            for (k = 0; k < dimension; k++) {
                auxDistance += (fireworkAndSparksPositions[i][k] - fireworkAndSparksPositions[j][k]) *
                    (fireworkAndSparksPositions[i][k] - fireworkAndSparksPositions[j][k]);
            }
            probabilityOfSelection[i] += Math.sqrt(auxDistance);
        }
        sumOfProbability += probabilityOfSelection[i];
    }

    double[] cumulativeProbability = new double[fireworkSparksNumber];
    for (i = 0; i < fireworkSparksNumber; i++) {
        if (sumOfProbability < EPS_VALUE) {
            probabilityOfSelection[i] = GAUSSIAN_COEFFICIENT_BASE / fireworkSparksNumber;
        } else {
            probabilityOfSelection[i] /= sumOfProbability;
        }
        cumulativeProbability[i] = (i == 0) ? probabilityOfSelection[i] : cumulativeProbability[i - 1] +
            probabilityOfSelection[i];
    }
    return cumulativeProbability;
}

```

Finalmente, se está en condiciones para seleccionar la mejor localización de la población, que será devuelta en forma de su spark contenedora (es reseñable recordar que las sparks son capaces de almacenar posiciones a tenor de su implementación):

```

private Spark selectBestLocation() {
    Spark bestSpark = fireworks[0];
    int i, j;
    for (i = 1; i < locationsNumber; i++) {
        if (fireworks[i].getValue(benchmarkFunction) < bestSpark.getValue(benchmarkFunction)) {
            bestSpark = fireworks[i];
        }
    }

    for (i = 0; i < locationsNumber; i++) {
        for (j = 0; j < sparks[i].length; j++) {
            if (sparks[i][j].getValue(benchmarkFunction) < bestSpark.getValue(benchmarkFunction)) {
                bestSpark = sparks[i][j];
            }
        }
    }
}

```

```

    }
  }
}
for (i = 0; i < gaussianSparksNumber; i++) {
    if (gaussianSparks[i].getValue(benchmarkFunction) < bestSpark.getValue(benchmarkFunction)) {
        bestSpark = gaussianSparks[i];
    }
}
optimumValue = bestSpark.getValue(benchmarkFunction);
return bestSpark;
}

```

Como inciso final, es importante introducir el método que define si el FA debe finalizar, en el que, para velar por la máxima calidad posible en los resultados, se asegura que se ejecute 10.000 veces:

```

private boolean stopCriteria() {
    if(numGenerations < 10000) {
        boolean success;
        if (numFunctionEvaluations < NUMBER_OF_FUNCTION_EVALUATIONS) {
            success = false;
        } else {
            System.out.println("NumGenerations=" + numGenerations + ",NumFunctionEvaluations=" + numFunctionEvaluations);
            success = true;
        }
        return success;
    }
}

```

Una vez explicada pormenorizadamente la clase fundamental del FA, se procederá someramente a hacer lo propio con *Main.java*, quedando el lector referido al repositorio GitHub si deseara consultar más aspectos de *FireworkAlgorithm.java*.

En aquella clase residen todos aquellos parámetros que, si bien no forman parte conceptual del FA, sí lo hacen en cuanto a la ejecución del mismo:

```

import java.io.FileNotFoundException;

/**
 * Created by Clemencio Morales Lucas.
 */

public class Main {

    public static final String INFORMATION_FILE_PATH_UNIX = "/home/clemen/FA_info.txt";
    public static final int NUMBER_OF_ITERATIONS = 20;
    public static final int LOCATIONS_NUMBER = 5;
    public static final int NUMBER_OF_SPARKS = 50;
    public static final double LOW_BOUND_NUMBER = 0.04;
    public static final double HIGH_BOUND_NUMBER = 0.8;
    public static final int MAXIMUM_AMPLITUDE_VALUE = 40;
    public static final int GAUSSIAN_SPARKS_VALUE = 5;
    public static final String AVERAGE = "Average values: ";

```

```

public static final String SHIFT = "|Shift ";
public static final String SEPARATOR = "|t";
public static final String FITNESS = "|Fitness ";
public static final String OUTPUT_SEPARATOR =
"===== ";
public enum HostSystem { WINDOWS, UNIX }

private double[] maximumBound;//= new double [30];
private double[] minimumBound;//= new double [30];
private final double[] shiftIndex = {0, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7};
private double[] availableBounds = {100, 100, 30, 32, 600, 5.12, 50, 5, 2, 100, 5.12, 65.536};
private int[] availableDimensions = {30, 30, 30, 30, 30, 30, 30, 2, 2, 2, 30, 30};
private String filePath;

```

Por otro lado, la presente clase, además de ejecutar el algoritmo en su método `launch()` y guardar la salida en un fichero de texto, inyecta el resto de variables que FA necesita en el siguiente método:

```

private void launchFWAForAllParameters() {
    BenchmarkFunctions[] benchmarkFunctionsSet = BenchmarkFunctions.values();
    for (int i = 1; i <= BenchmarkFunctionConstants.NUMBER_OF_FUNCTIONS; i++) {
        System.out.print(FITNESS + "-" + benchmarkFunctionsSet[i-1] + "-" + i + SEPARATOR);
        System.out.println();
        maximumBound = new double[availableDimensions[i - 1]];
        minimumBound = new double[availableDimensions[i - 1]];
        for (int j = 0; j < maximumBound.length; j++) {
            maximumBound[j] = availableBounds[i - 1];
            minimumBound[j] = -availableBounds[i - 1];
        }
        for (int k = 0; k < shiftIndex.length; k++) {
            BenchmarkFunction benchmarkFunction = new BenchmarkFunction();
            benchmarkFunction.setIndexAndShift(i, availableBounds[i - 1] * shiftIndex[k]);
            double avg = 0;
            for (int t = 0; t < NUMBER_OF_ITERATIONS; t++) {
                FireworkAlgorithm fireworkAlgorithm = new FireworkAlgorithm(LOCATIONS_NUMBER,
                NUMBER_OF_SPARKS,
                LOW_BOUND_NUMBER, HIGH_BOUND_NUMBER, MAXIMUM_AMPLITUDE_VALUE,
                GAUSSIAN_SPARKS_VALUE,
                maximumBound, minimumBound, filePath, benchmarkFunction);
                avg += fireworkAlgorithm.launch();
            }
            avg /= NUMBER_OF_ITERATIONS;
            System.out.print("\t" + avg);
        }
        System.out.println();
    }
}

```

Finalmente, el algoritmo se ejecuta en el propio método `main`. Es fundamental hacer un inciso para destacar la existencia de una fase de calentamiento del hardware, en la que el algoritmo se ejecuta 2000 veces antes de capturar los resultados, así como se desactivan todos aquellos elementos, a través del IDE, que pudieran interferir con los resultados en cualquier sentido. De esta forma, se obtienen unos resultados libres de ruido y fiables:

```

public static void main(String[] args) throws FileNotFoundException {
    System.out.println("-----INITIALIZING WARMUP PHASE-----");
    for (int i = 0; i < 2000; i++) {
        System.out.println("+++++++ "+i+" ++++++");
        new Main().launch(HostSystem.UNIX, false);
    }
    System.out.println("-----HARDWARE WARMUP PHASE ENDED-----");

    HostSystem hostSystem = HostSystem.UNIX;
    long initialTime = System.nanoTime();

    new Main().launch(hostSystem, false);

    long finalTime = System.nanoTime();
    long elapsedTime = finalTime - initialTime;

    System.out.println(OUTPUT_SEPARATOR);
    System.out.println("Time elapsed for current algorithm (ns): "+ elapsedTime);
    System.out.println(OUTPUT_SEPARATOR);
}

```

Finalmente, la ejecución del algoritmo se refleja a continuación:

Figura XIII: Resultado de la ejecución del Firework Algorithm clásico implementado en la presente Tesis Doctoral

Tras haber descrito FA al mayor nivel de detalle y pormenorización posible, así como haber presentado a lo largo del presente epígrafe sus más importantes características y operadores, queda evidenciada la enorme potencia algorítmica de la entidad.

Concluyendo el presente epígrafe, por ende, se destaca que FA y sus variantes han demostrado con creces que pueden resolver cualquier problema de optimización compleja de manera no sólo eficaz (consiguiendo todos los

objetivos) sino eficiente (consiguiendo todos los objetivos de la mejor manera posible). Por si esta deseable característica se antojara insuficiente, FA puede ser ejecutado en paralelo, siendo susceptible a aplicarse en problemas de Big Data u otros donde el tamaño de la población sea hiperbólico, lo que lo sitúa en una posición privilegiada como un algoritmo formidable.

2.1.2 Genetic Algorithms

Tal y como postula Charles Darwin (1809-1882) en su opus magnum *El origen de las especies* [Darwin,1859], desde tiempos inmemoriales los seres vivos se han visto forzados a un continuo proceso evolutivo en pos de su supervivencia. Toda especie de ser vivo que se precie, evoluciona a partir de un antecesor común con el fin de adaptarse a su entorno y sobrevivir, llevando a cabo el proceso denominado selección natural. Numerosas teorías y razonamientos en la misma línea defienden el mismo argumento, tales como la comúnmente denominada *Teoría de la Jirafa de Lamarck* [Lamarck, 1801], que toma su nombre del naturalista Jean Baptiste Lamarck. La anteriormente citada Teoría establece que la longitud del cuello de la jirafa viene determinada por la necesidad, directamente proporcional, de dicho animal a alcanzar las hojas de las elevadas acacias africanas. Este comportamiento evolutivo, motivado por la necesidad de supervivencia intrínseca a los seres vivos, se refleja fielmente en la siguiente figura:

Figura XIV: Teoría de la Jirafa de Jean Baptiste Lamarck (1801)

Tal y como se especificará a lo largo del presente epígrafe, la naturaleza de los Algoritmos Genéticos (*Genetic Algorithms*) [Holland, 1975] hará del principio anteriormente descrito su característica atómica. Con el fin de situar al lector en la temática que nos ocupa, se presenta el siguiente relato introductorio, que servirá para establecer las bases de la algorítmica de corte genético:

"Mucho tiempo atrás, existía un banco de peces prehistóricos que vivían desde hace eones en una oscura cueva subacuática. Dado que la luz solar no tenía

cabida en su ecosistema, el primitivo sistema sensorial de los peces no contaba en su haber con ninguna suerte de órgano visual, siendo su olfato el único elemento necesario para ellos a fin de alimentarse del plancton que surgía de las paredes de la cueva. Un buen día, la vibración provocada por el movimiento de las placas tectónicas abrió una grieta en la cueva, por la que los peces salieron, tras un tiempo inenarrable, a la búsqueda de alimento fuera de la misma. Además de nuevos tipos de alimento, los peces encontraron una agresiva especie depredadora formada por escualos, que mermaron la población de pescados de la cueva. La suerte de esta población, en jaque, cambió su negro destino en la siguiente generación de peces, en la que algunos individuos desarrollaron una mutación; una suerte de colmillos poblaron su mandíbula para alimentarse de una mejor manera de los nuevos tipos de alimento presentes en el exterior de la cueva, mientras que unos ojos primitivos surgieron en sus cabezas. De esta forma, los peces mutados incrementaron notablemente su probabilidad de supervivencia, pues podían alimentarse de un mayor rango de alimentos, así como detectar la presencia de tiburones fuera de la cueva. Tan importante ventaja presente en ciertos individuos del banco de peces, atrajo la preferencia por las hembras a la hora del apareamiento, por lo que los colmillos y los ojos se propagaron a las sucesivas generaciones, salvando así la especie de su aparentemente anunciada extinción."

En el anterior relato, se ha podido observar cómo una población de individuos (peces) se ve forzada a evolucionar como resultado de una acción aleatoria en su entorno (*apertura de la cueva por un temblor*). Para ello, una serie de mutaciones (*nuevos colmillos y ojos*) acontecen, así como una selección (*actitud preferente de las hembras hacia los peces mutados*). Como resultado del anteriormente citado proceso de selección, ciertos individuos de la especie estarán más adaptados a su ecosistema, sobreviviendo por ende, mientras que, para su desgracia, los especímenes que no hayan logrado adaptarse se verán destinados a su desaparición. Este es, precisamente, el principio de funcionamiento de un Algoritmo Genético:

Se define como Algoritmo Genético (Genetic Algorithm; GA en lo sucesivo) a aquel conjunto de instrucciones ordenadas, en pos de la resolución de un problema específico, que se basan en la base genético molecular del proceso evolutivo de los seres vivos. Esta rama de la algorítmica, adscrita al Paradigma de la Computación Natural, surge de la mano de las investigaciones del profesor John Henry Holland (1929-2015), pionero en la investigación de los Algoritmos Genéticos, en la década de 1970. A pesar de que se atribuye la paternidad de los GA al profesor Holland, su excelso trabajo supone el colofón de un ciclo investigador iniciado por el insigne Gregor Mendel en 1865, con las *Leyes de Mendel* basadas en la investigación del *Pisum Sativum* o Guisante Común. Mendel describe, utilizando dicha variedad de guisantes, las principales reglas básicas de la transmisión de características entre individuos mediante la herencia

genética. Cabe destacar, como dato anecdótico, que Gregor Mendel engrosa la lista de autores de diversos campos que, desgraciadamente, nunca llegaron a disfrutar de la popularidad de sus trabajos debido a su fallecimiento prematuro, en 1884. No fue hasta 1900 cuando las Leyes de Mendel, redescubiertas, gozaron de la popularidad que merecían, por lo que Gregor Mendel pasa a engrosar tan tristemente amplio *ranking*, entre los que se encuentran el epígono del Romanticismo español Gustavo Adolfo Bécquer o el actor norteamericano Heath Ledger, por citar algunos.

Las Leyes de Mendel establecen, principalmente, los procesos de transmisión de características de un progenitor a su descendencia. En la clase de guisantes que Mendel empleó, detectó ciertos caracteres, de tipo dominante, que se caracterizaban por determinar el efecto de un gen, y recesivos, catalogando por estos últimos aquellos que carecían de efecto sobre el genotipo heterocigótico. La siguiente figura ilustra el Cruzamiento Monohíbrido Mendeliano entre un *Pisum Sativum* de superficie alisada y un homólogo de piel rugosa:

Figura XV: Cruzamiento Monohíbrido Mendeliano entre dos *Pisum Sativum*

La Teoría Mendeliana supuso un inestimable punto de apoyo para la considerada como la primera *Teoría de la Evolución Biológica* [Lamarck, 1809] de Jean Baptiste Lamarck. En dicha Teoría, contenedora de la *Teoría de la Jirafa* introductoria al presente ámbito, se razona que los seres vivos no han sido concebidos desde un punto de vista *creacionista*, sino que evolucionan de forma periódica a partir de organismos primitivos. La Teoría en cuestión defiende, además, que los seres vivos varían adaptándose a su entorno y circunstancias, contrastando dicha argumentación con los fósiles encontrados hasta la fecha. De esta forma, Lamarck pretendía demostrar, de manera empírica, que tales fósiles no eran sino las reminiscencias de organismos intermedios entre la fauna contemporánea a su época y las formas de vida primigenias.

Aunando de manera brillante las Teorías anteriormente citadas, *Charles Darwin* afirma, en su obra *El origen de las especies* [Darwin et al., 1859], que "... *One general law, leading to the advancement of all organic beings, namely, multiply, vary, let the strongest live and the weakest die.*" ("*Una ley general, que conduce a la evolución de todos los seres orgánicos, es decir, multiplicarse y variar, deja al más fuerte vivir y al más débil fenecer.*")

Como se relatará a continuación, John Henry Holland, con la inestimable ayuda de su brillante pupilo David Edward Goldberg, cierran el círculo iniciado por Gregor Mendel en 1865 extrapolando la Teoría Genética *Universal* al ámbito computacional mediante los GA. Así, un interrogante puede asaltar de forma natural al lector, y es que, ¿cuál es exactamente la función de un GA?

Un GA tiene como objetivo principal hacer evolucionar a ciertos especímenes que componen una *población*. Para ello, utiliza acciones de corte aleatorio que establecen un símil con los procesos naturales de la evolución biológica. Estos métodos, denominados como *operadores genéticos*, se relatan a continuación:

- Selección (Selection): Proceso mediante el cual se eligen ciertos individuos a partir de una población, con la finalidad de ser evolucionados o reinsertados en la población una vez evolucionados. Existen cuatro tipos diferentes de selección, que definen su taxonomía por consiguiente:

1. Selección de Ruleta (*Roulette-Wheel Selection*) [Holland, 1975] o Selección Proporcional a la Función de Desempeño (*Fitness Proportionate Selection*): La presente metodología genética de selección establece que la probabilidad de que un individuo sea seleccionado se calcula por la siguiente fórmula, donde N es el número de individuos existentes y f_i es el desempeño del i -ésimo individuo:

$$p_i = \frac{f_i}{\sum_{j=1}^N f_j},$$

El presente modelo de selección provoca que los mejores individuos sean escogidos con una probabilidad mayor, mientras que a su vez permite que los peores individuos sean elegidos, aunque con una probabilidad menor. De esta manera, se mantiene la diversidad de la población, en contraposición con la Selección por Truncamiento que se describirá seguidamente. El principal problema de la Selección de Ruleta aparece en el escenario en el que una pequeña parte de la población posee una medida de desempeño excesivamente superior al resto. Tal casuística provoca la pérdida de diversidad, desembocando de forma irremediable en

la convergencia prematura de la iteración, ya que la mayoría de los individuos seleccionados serán una copia idéntica de los pocos seres predominantes. En tal caso, se aconseja encarecidamente utilizar el proceso de Selección Basada en Ranking o la Selección por Torneo.

2. Selección por Truncamiento: En la presente modalidad de selección, las soluciones candidatas se ordenan en base a su *fitness function*, mientras que una proporción de los individuos con mejor aptitud será seleccionada y reproducida $1/p$ veces. Dado el carácter primitivo de esta modalidad de selección, se desaconseja su uso práctico.

3. Selección basada en *Ranking* [Baker, 1985]: En la presente modalidad de selección, los individuos son ordenados a tenor de su función de aptitud en un *ranking*. Una vez hecho, son asignados con una *medida de desempeño*, directamente proporcional a su posición en el *ranking*. Se destaca que los valores de esta segunda asignación pueden oscilar en el ámbito lineal o exponencial. Por último, los individuos son seleccionados proporcionalmente a tal probabilidad. De esta manera, se otorga una mayor probabilidad de selección a los mejores individuos de la población. El método de Selección basada en *Ranking* consigue reducir el riesgo de convergencia prematura que puede acontecer cuando se usa la Selección basada en Ruleta en ciertas poblaciones, en las que unos pocos individuos constan de medidas de aptitud atípicas (en cuanto a su superioridad) entre sus congéneres.

4. Selección por Torneo [Miller et al., 1995]: En este último tipo de selección, se realiza un torneo o comparación entre un pequeño número de individuos elegidos al azar de entre la población. La simplicidad en cuanto a su diseño acelera enormemente el tiempo de computación, puesto que no se necesita evaluar la población en su completitud. Esta característica, unida a la capacidad de prevenir la convergencia prematura hasta cierto punto, se corresponde a la principal ventaja de la selección por torneo. En contraposición, la desventaja fundamental de esta modalidad de selección es la dificultad de establecer cuál es el parámetro correspondiente al tamaño del subconjunto de tal manera que se evite, en un rango aceptable, la posibilidad de convergencia prematura, que sigue existiendo a pesar de ser indudablemente menor.

- Cruzamiento (*Crossover*) [Janikow et al., 1991]: Se conoce como cruzamiento al proceso mediante el cual se genera una variación en los alelos de un cromosoma desde una generación a la siguiente. Es reseñable que el cruzamiento constituye una analogía con la reproducción sexual de los seres vivos. Tomando como referencia un conjunto de cromosomas representados como cadenas binarias de información, en las

cuales se ahondará en breve, destacan las siguientes metodologías de cruzamiento:

1. Cruzamiento en un punto: Una vez seleccionado un punto en el primer cromosoma dado, todos los bits o alelos sucesivos se intercambiarán entre dicho cromosoma y su pareja, generando una nueva descendencia. La siguiente figura ilustra el sistema de cruzamiento en un punto:

Figura XVI: Esquema de la técnica de Cruzamiento o *Crossover* en un punto

2. Cruzamiento en dos puntos: De forma muy similar al cruzamiento anteriormente descrito, en este caso se seleccionan dos intervalos entre los cromosomas primitivos, que serán intercambiados dando lugar a los cromosomas descendientes:

Figura XVII: Esquema de la técnica de Cruzamiento o *Crossover* en dos puntos

3. Cruzamiento *de corte y empalme*: Esta técnica de cruzamiento, que desemboca en un cambio en la longitud de los cromosomas generados, consiste en intercambiar el sobrante de los individuos originales tras efectuar un "corte", tal y como muestra la siguiente figura:

Figura XVIII: Esquema de la técnica de Cruzamiento o *Crossover* corte y empalme

4. Cruzamiento uniforme: Técnica de cruzamiento mediante la cual los bits del cromosoma se comparan de forma individual entre ambos padres, intercambiándose con una probabilidad de, normalmente, 0.5:

Figura XIX: Esquema de la técnica de Cruzamiento o *Crossover* uniforme

5. Cruzamiento uniforme medio [Eiben, 1994]: En esta variación del esquema anterior, únicamente la mitad de bits distintos entre las cadenas de cromosomas se intercambian. Para ello, se calcula el número de bits que difieren mediante el Teorema de la Distancia de Hamming, para posteriormente dividir el número resultante entre dos. Dicho número se corresponde con la cantidad de bits que han de ser intercambiados entre los cromosomas originales, dando lugar a una nueva descendencia:

Figura XX: Esquema de la técnica de Cruzamiento o *Crossover* uniforme medio

6. Cruzamiento de cromosomas ordenados: En caso de que la metodología de representación de los cromosomas no sea en formato binario, el cambio directo podría ser imposible. Esta circunstancia ocurre cuando existe un determinado orden en los elementos que componen el cromosoma, escenario en el cual el sub-cromosoma hasta el punto de cruzamiento seleccionado se retiene para cada progenitor. Posteriormente, la información se cruza y se ordena de la forma original en la que estaba en el otro progenitor.
- Mutación: Variación en el genotipo (información de tipo genética) de un ser vivo. La variación anteriormente descrita, que puede transmitirse a la

descendencia del individuo, se presenta motivada por la acción de mutágenos en el ecosistema. Se destaca llegado este punto que la unidad genética capaz de mutar es el gen, unidad de información atómica y hereditaria que compone el ADN del individuo.

- **Recombinación:** Proceso mediante el cual una porción de ADN se escinde para posteriormente unirse a una molécula de material genético distinto. Este comportamiento produce combinaciones genéticas diferentes en la especie con respecto a sus predecesores, con lo que se producen alelos quiméricos. Esta ventaja permite efectuar la reproducción sexual a los seres vivos, así como evitar el Trinquete de Muller, correspondiente al proceso por el cual los genomas acumulan mutaciones perjudiciales de forma irrevocable.

Realizando una escisión en el carácter de los GA, destacan varias partes claramente diferenciadas, muchas de las cuales han aparecido de forma previa en el presente epígrafe, pero cuyos conceptos conviene asentar: En primer lugar, los GA parten de una población de individuos que pueden representarse como una cadena binaria, denominada cromosoma. Cada uno de estos elementos estará constituido por un conjunto de genes, que a su vez se compondrán de alelos. Dicha población de individuos se denomina genotipo:

Figura XXI: Estructura atómica de un cromosoma

El conjunto de cromosomas o población evolucionará a través de *iteraciones* o *generaciones*, en las que serán evaluados usando una función de aptitud (*fitness*). Las siguientes generaciones de cromosomas se generarán mediante la aplicación repetida de los operadores genéticos, de tal forma que, eventualmente, se llegará a alcanzar el conjunto de soluciones del problema, denominado *fenotipo*.

Alcanzado el presente punto, se está en disposición de atender a la descripción genérica del funcionamiento de un GA, que consta de las siguientes fases:

- **Inicialización:** La población inicial es generada en base a un esquema aleatorio típicamente. Cabe destacar que esta inicialización no es siempre posible en función de las características del problema a solucionar, siendo un escenario explicativo aquel en el que se pretende hallar el camino mínimo entre dos puntos de un grafo. En caso de que la inicialización no sea aleatoria, se debe garantizar que, dentro del genotipo, exista una aceptable diversidad de cromosomas o soluciones, a fin de representar fielmente la totalidad de la población existente y con el objetivo principal de evitar la condición prematura.
- **Evaluación:** Se aplica la *fitness function* o función de aptitud a todos los elementos de la población. De esta manera, se cuantificará el valor de cada individuo de la población.
- **Finalización o Condición de Término:** Debe acontecer cuando se alcance la perfección en un individuo de la población, que pase de genotipo a fenotipo. Esta solución óptima se desconoce en la mayoría de los casos, por lo que deben utilizarse criterios de respaldo. Dichos criterios son, a saber: Ejecutar el GA un número lo suficientemente elevado de iteraciones o bien detenerlo cuando no haya cambios en el genotipo. En caso de que ninguno de los anteriormente citados criterios se presente, se deben aplicar de forma iterativa los operadores genéticos detallados previamente en el presente epígrafe.

La figura que se presenta seguidamente engloba y esquematiza el proceso de funcionamiento genérico de los GA anteriormente razonado, donde *i* representa la fase de inicialización, *f(X)* la de evaluación, ? la bifurcación referente a la condición de terminación, *Se* la selección, *Cr* el cruzamiento, *Mu* la mutación, *Re* el reemplazo y *X** la mejor solución:

Figura XXII: Funcionamiento genérico esquematizado de un Algoritmo Genético

Dada la vital importancia de los GA en la presente Tesis Doctoral, se presenta el siguiente diagrama de flujo de los GA, que complementa la figura anterior:

Figura XXIII: Diagrama de flujo genérico de un Algoritmo Genético

Una vez expuesto el funcionamiento de un GA; ¿Cuándo pueden aplicarse este tipo de algoritmos? La respuesta a tal cuestión tiene un amplio número de soluciones, pues como ya se ha especificado en el presente documento, una de las ventajas de los GA es que sus escenarios de aplicación se encuentran limitados, casi únicamente, por la imaginación. Cabe citar, como ejemplos, la construcción de árboles filogenéticos, el diseño de topologías para circuitos impresos y las redes computacionales, la resolución de equilibrios en la Teoría de Juegos, el aprendizaje automático en el comportamiento robótico, la Lógica Difusa, el procesamiento de lenguajes naturales, la predicción del plegamiento de proteínas y, por supuesto, en la ingeniería del software, entre muchos otros. Como ejemplo concreto de la aplicación de algorítmica de corte genético, se presenta un software que, con casi total seguridad, será familiar para el lector amante de los videojuegos: *Quake III Arena*. En este popular videojuego, de tipo shooter, un grupo de *robots software*, denominados *bots*, compiten entre sí en base a una lógica difusa [Berkeley, 2001]. Dicha competición consiste en una suerte de *western* futurista, en el que los jugadores dispondrán de armas, tecnológicamente avanzadas, para eliminar al enemigo:

Figura XXIV: Fotograma de Quake III Arena

El jugador humano puede competir contra otros seres humanos y contra los *bots*, siendo este el punto donde los GA aparecen a colación: En la generación de estos agentes *software*, se inicializa el GA creando, aleatoriamente, un conjunto de *bots*, que parten de la misma lógica difusa. Una vez estos han competido entre sí, se ordena a los mismos en un *ranking*, que constituye la función de aptitud o *fitness* al reflejar el ratio de victorias y derrotas de cada bot. En función de dicho *ranking*, se aplicará el cruel pero eficaz principio de la selección natural, que se recuerda consiste en descartar los individuos de peor rendimiento. Una vez hecho, se asignará una probabilidad de reproducción a los *bots* que será inversamente proporcional a su posición en el *ranking*. Por tanto, para cualquier bot del sistema, su probabilidad de reproducción será mayor cuanto mejor sea su clasificación en el *ranking*. En este momento, los *bots* restantes se mezclarán entre sí, usando la media aritmética entre los valores difusos de las relaciones de ambos progenitores como operador genético de reproducción. Estas operaciones, de carácter difuso, se cruzarán, alteradas (mutadas) aleatoriamente con el fin de concebir relaciones borrosas del nuevo *bot*. El proceso será nuevamente ejecutado, instigando a los *bots* a competir de nuevo para crear un nuevo *ranking* entre individuos mejorados. Este *ranking* desembocará en el conjunto de *bots* a los que se enfrentará el jugador, siendo el número de iteraciones proporcional al nivel de dificultad elegido por el jugador.

Por otro lado, en términos de efectividad y a pesar de la efectividad de los GA, es bien sabido que no suele existir una infalible *bala de plata* en el ámbito de la ingeniería informática [Wolpert et al., 1995], y, desafortunadamente, los GA no

son una excepción. Entre las limitaciones principales de los GA pueden encontrarse las siguientes características perniciosas:

- Un problemático escenario en el ámbito de los GA es aquel en el que los datos de entrada están dotados de un carácter dinámico [Banzhaf et al., 1998]. En ellos, es posible que, en la búsqueda del fenotipo, el genotipo converja a soluciones que dejan de ser válidas dada la mutabilidad de la información. Una de las posibles soluciones a este problema parte de la inclusión de la hipermutación, consistente en el incremento de la probabilidad de mutación cuando la igualdad de la solución disminuye. Otra alternativa consiste en introducir inmigrantes aleatorios, correspondientes a cromosomas aleatorios en el genotipo.
- En un gran número de problemas, los GA presentan una tendencia, difícil de frenar, a converger hacia máximos locales y no globales [Bies et al., 2006]. Este comportamiento provoca que el punto óptimo global del fenotipo no sea alcanzado. Como ejemplo de la presente problemática, considérese la siguiente abstracción, consistente en un problema de optimización aproximado por GA extrapolado a un ejemplo del mundo real en el que un escalador, situado al inicio de una ladera con los ojos vendados, debe trepar una montaña. La ayuda de sus manos mediante la palpación del entorno es su único elemento para saber si se encuentra en el pico de la montaña. En este escenario, suponiendo que la escalada comienza desde la esquina inferior izquierda de la imagen, es posible identificar el problema con celeridad:

Figura XXV: Problemática de los GA en la búsqueda de óptimos globales en el espacio de búsqueda

Tal y como puede extraerse del examen de la figura, cuando el escalador alcance el punto óptimo local (*local optimum*) pensará que está en la cima,

pues no palpará ningún punto más alto. Este razonamiento es evidentemente incorrecto, pues el punto más alto (*peak*) se encuentra a continuación y su búsqueda no debería finalizar.

El anteriormente ejemplo pone de relevancia la dificultad de conocer cómo sacrificar la eficacia a corto plazo del algoritmo en pos de lograr un mejor fenotipo. Este problema, cuya solución no es trivial, puede aliviarse mediante el uso de funciones de aptitud diferentes, incrementando el *ratio* de mutación, o bien mediante el uso de técnicas de selección que incrementen el ratio de mutación. De la misma manera, dichas técnicas podrían, idealmente, mantener una diversidad aceptable en el genotipo, a pesar de que el *No Free Lunch Theorem* demuestra empíricamente que no existe una solución general para este problema. Una técnica común para mantener la diversidad es la de imponer una *penalización de nicho* en aquellos cromosomas similares dentro del genotipo. De esta forma, se disminuirá la probabilidad de que dichos individuos se *propaguen* hacia nuevas generaciones, permitiendo que los individuos más dispares se mantengan.

- La repetición de la *fitness function* o función de aptitud para problemas de índole compleja es, con frecuencia, el segmento más prohibitivo de los GA. Encontrar la mejor solución de problemas multidimensionales, requiere funciones de aptitud extremadamente prohibitivas desde el punto de vista computacional [Cha et al., 2009]. Con el fin de otorgar al lector un ilustrativo ejemplo, baste considerarse los problemas reales de optimización estructural, donde una única función de aptitud para un cromosoma dado puede necesitar horas (e incluso días) en completarse. Por consiguiente, los métodos de optimización clásicos son insuficientes para lidiar con los problemas adscritos a la taxonomía en cuestión. En tal escenario, existe la necesidad de realizar una aproximación a la función de aptitud, de manera que esta sea aceptable desde el punto de vista del tiempo computacional. Es reseñable que la amalgama de metodologías que pueden surgir de dicha necesidad se corresponde con una de las aproximaciones más comprometedoras en la actualidad, que idealmente desembocará en el uso de los GA para resolver problemas existentes en el mundo real.
- Los GA presentan un acuciante problema en cuanto a su escalabilidad en función de la complejidad. En otras palabras, cuando el genotipo es excesivamente numeroso, el espacio de búsqueda se incrementa siguiendo una distribución exponencial [Goldberg, 1989]. Esta perniciosa característica prácticamente impide el uso de los GA en aquellos problemas donde las posibles soluciones tienden al infinito, como es el caso de la antena para el satélite ST5 de la NASA presentado en este

escrito, el diseño de motores de manera informatizada, el diseño de edificios y problemas de analogía similar. La única manera de sobreponer el elevado coste computacional en este escenario es la de contar con numerosos equipos de altas prestaciones, pudiendo necesitarse incluso el uso de un supercomputador. Por otro lado, la representación de los datos adscritos al problema deben simplificarse al máximo. En la misma línea, otro de los grandes problemas referidos a los GA es la complejidad referente a cómo implementar una protección de los subconjuntos deseados en un cromosoma frente a su variación a la hora de aplicar los operadores genéticos. Cuando un cromosoma complejo varía en una iteración, es altamente probable que ciertos alelos transiten al estado deseado, mientras que lo contrario ocurra a sus homólogos. En la siguiente iteración, el azar podría conseguir que la aplicación de los operadores genéticos podría variar de nuevo el estado de los alelos deseados, lo cual provocaría un impacto negativo en el rendimiento del algoritmo. El evitar este proceso tiene una elevada importancia, directamente proporcional a su complejidad, especialmente en problemas en los que la función de aptitud necesita combinar los cromosomas que forman el genotipo.

- La considerada como mejor solución lo es únicamente desde un punto de vista subjetivo, pues lo es sólo en comparación con otras soluciones. Por consiguiente, el criterio de finalización o parada es difuso en la amplia mayoría de problemas que los GA pretenden resolver.
- En cuanto a los problemas de optimización específica, que podrían estar relacionados con ciertas instancias, otro modelo de optimización podría ser más eficiente que los GA en términos de velocidad de convergencia hacia el fenotipo. La Computación en Enjambre, la Optimización basada en Colonia de Hormigas y la Optimización por Enjambre de Partículas, que serán detalladas en sucesivos epígrafes de la presente Tesis Doctoral, son algunos de los modelos de optimización alternativos. Es reseñable que el grado de idoneidad de los GA depende a su vez de la cantidad y la calidad del conocimiento del problema, pues los problemas frecuentes y conocidos pueden constar de aproximaciones de resolución más certeras.
- Los GA son prácticamente incapaces de solucionar problemas donde la única función de aptitud se corresponde con una decisión *booleana*, pues no hay forma de converger hacia la solución. Esta característica es fielmente reflejada por el ejemplo del escalador expuesto en el presente epígrafe. En escenarios análogos, una búsqueda simple de corte aleatorio puede encontrar una solución con un rendimiento similar a un GA, pero con una menor complejidad dada la mayor enjundia inherente a la programación genética (Goldberg, 2002). En cualquier caso, si la

naturaleza del problema permite la aplicación de la técnica Prueba y Error dando lugar a diferentes resultados, el radio de éxitos por fallo (que puede ser modelado estadísticamente por una Distribución de Poisson al contar un número de eventos por unidad) puede otorgar una aceptable función de aptitud.

Conocidas las limitaciones de los GA, es recomendable reseñar que existen ciertas variaciones en cuanto a los mismos. Dichas variaciones se aplican, principalmente, en casos en los que el carácter del problema a resolver por el GA está impregnado de un carácter atípico.

Tal y como se ha enfatizado a lo largo del presente escrito, el GA típicamente trabaja con un conjunto de datos de entrada que representan una población de especímenes. Esta población, denominada cromosoma, típicamente está representada en un formato binario. De esta manera, el cromosoma consta de una naturaleza numérica, que, en términos informáticos, puede ser representada como un valor *Integer* o *Float*. A pesar de que este es el escenario típico para un alto porcentaje de problemas a resolver por la algorítmica genética, en el que las operaciones genéticas se aplican a nivel de alelo o bit, existen ciertas variaciones. En ciertos casos, los GA requieren de la representación del cromosoma como una lista numérica indexada en función de una tabla de instrucciones, nodos adscritos a una lista enlazada [Fogel, 2003], estructuras de tipo *hash*, o, en su versión más genérica, objetos del paradigma POO (Programación Orientada a Objetos; Object Oriented Programming). En estos casos, las operaciones genéticas deben variarse en pos de su aplicación con restricciones, lo que presenta un handicap añadido al GA; no sólo se debe cubrir la complejidad algorítmica del problema, sino que a su vez se debe respetar las directrices que establecen el orden de los datos. Tal necesidad desemboca en la sobreescritura, en la mayoría de los casos, de los operadores genéticos, que gozarán de una mayor y más compleja funcionalidad para tratar con el cromosoma especial. Un ejemplo ilustrativo de este escenario, es el caso en el que se pretende aplicar una metodología de resolución basada en GA a un problema de búsqueda de rutas, en el que los itinerarios podrían estar, por ejemplo, codificados en formato de grafo. En este caso, las operaciones genéticas deben ser especialmente cuidadosas, pues la simple mezcla aleatoria del conjunto de vértices y aristas que componen el grafo provocaría un cambio en la ruta original, que sería destruida.

Por otro lado, es digno de mención que algunas aproximaciones de representación atípica del cromosoma involucran el uso de vectores decimales en lugar del típico conjunto de alelos binarios. Aunque en un principio pueda parecer que este cambio impacta negativamente en el rendimiento del algoritmo (el cardinal del alfabeto utilizado es inversamente proporcional al rendimiento del GA), esto no ocurre en todos los casos. Ocasionalmente, el conjunto de valores

reales en la población discreta de cromosomas componen un alfabeto virtual, con una cardinalidad mucho menor de la que podría esperarse de una representación en base a valores de tipo *Float*.

En caso de que no sea necesario adoptar ningún tipo de esquema especial para el tratamiento del cromosoma, se recomienda encarecidamente codificar la cadena de alelos mediante el Código Gray. Esta metodología de representación numérica en base dos, establece la premisa de que dos valores sucesivos difieren únicamente en un bit. A continuación se presenta la tabla de conversión entre código decimal, binario y Gray:

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

Figura XXVI: Tabla de conversión entre el Sistema Decimal, Binario y Gray

Como puede intuirse, el uso del Código Gray viene motivado por la ligera variación que este provoca en el valor numérico; dado que cambios ligeros en el valor numérico pueden variarlo fácilmente por la aplicación de las operaciones genéticas, el Código Gray ayuda a evitar la convergencia prematura del cromosoma.

Finalmente, en cuanto a la representación de cromosomas se refiere, es reseñable que otra de las líneas alternativas de formato parte de la concepción del cromosoma como un conjunto de genes heterogéneos. De esta forma, se posibilita la resolución de problemas de optimización donde se requiere una extensa definición de dominios para la población de especímenes, en la que cada individuo está compuesto de secciones bien diferenciadas. En este caso, y una vez más, los operadores genéticos deberán ser reescritos de forma específica para lidiar con los pormenores del problema concreto, así como sus directrices especiales.

Continuando en el ámbito de las variaciones existentes en los GA, los GA Adaptativos (Adaptive Genetic Algorithm; AGA en lo sucesivo) interpretan un rol

fundamental. Dado que las probabilidades de cruzamiento y mutación determinan en gran medida la precisión de la solución propuesta por un GA, así como el rendimiento, entendiendo como tal la celeridad en la convergencia del genotipo hasta el fenotipo, los AGA utilizan una variación del cruzamiento y la mutación. En esta variante, los algoritmos utilizan la información del cromosoma en cada generación, y la adaptan de forma dinámica en caso de que se requiera asegurar la diversidad de los especímenes. A su vez, esta característica mantiene una aceptable velocidad de convergencia. En los AGA, el ajuste de las características asociadas al cruzamiento y la mutación está intrínsecamente relacionado con la función de aptitud, que deberá ajustarse a dichos operadores genéticos en función de la naturaleza del problema a resolver.

Otro tipo de GA adaptativos se corresponden con los Algoritmos Genéticos basados en Racimo (*Clustering-based Adaptive Genetic Algorithms*), CGA en lo sucesivo. Los CGA destacan, como su propio nombre indica, por el uso del análisis en cluster para determinar los estados en los que el cromosoma es óptimo, de los que dependerán, una vez más, los operadores genéticos. Como se evidenció de forma previa con el problema del escalador, los GA presentan una capacidad aceptable para encontrar soluciones aceptables, pero en ocasiones son ineficientes para encontrar la solución óptima absoluta. De esta manera, se recomienda la implementación de sistemas de visualización panorámica, donde, una vez el algoritmo crea haber encontrado la mejor solución, examine independientemente el siguiente tramo del cromosoma para contrastar la solución encontrada y determinar, empíricamente, si es la mejor. La dificultad principal de esta metodología reside en la complejidad de establecer cuál es la "distancia" a examinar una vez encontrada una solución, de cara a aunar la efectividad del algoritmo con su rendimiento. Es digno de mención que una alternativa a la solución anteriormente descrita es la de tratar al genotipo como un todo, en la que los operadores genéticos actuarán a nivel de genotipo y no a nivel de cromosoma, como es frecuente. Esta variante se conoce como *Recombinación de Conjunto de Genes* (*Gene Pool Recombination*).

Como cierre de la línea de variaciones adscritas a los GA, y una vez detalladas las mismas en función de la representación de los datos, se presenta la variante del *Elitismo Genotípico*. En este ámbito, la principal característica de los GA Elitistas es la de permitir al mejor cromosoma del genotipo avanzar a la siguiente generación sin sufrir ninguna alteración o mutación en sus genes. La permanencia de todos sus alelos garantiza que la calidad de la solución obtenida por el GA no decrementará entre generaciones, a la vez que introduce una celeridad extra en la ejecución del algoritmo. Precisamente en términos de velocidad de ejecución del GA, es reseñable que, como en la gran mayoría de problemas del ámbito de la ingeniería informática, la computación concurrente tiene cabida en dos variantes [Larrañaga, 1996]:

- GA Paralelos de Grano Grueso: Esta rama algorítmica, cada subconjunto del genotipo residirá en un nodo o thread, de tal forma que se posibilitará el intercambio de cromosomas entre nodos.
- GA Paralelos de Grano Fino: En contraposición a la técnica anterior, y como el lector intuirá, en esta variante cada cromosoma disfrutará de un nodo dedicado. Cada conjunto de genes interactuará con sus homólogos cercanos para llevar a cabo las operaciones genéticas.

De forma previa a la presentación de un GA de ejemplo, procede recordar las características principales de un GA clásico:

- No debe perderse de vista que un GA es, de forma atómica, una función de optimización [Akbari, 2010]. Por definición, tal función tiene como principal objetivo hallar $(x_i, x_j, \dots, x_n)$ tales que $F(x_i, x_j, \dots, x_n)$ alcance el valor máximo.
- Una vez parametrizado el problema a resolver, las variables de entrada al GA compondrán un genotipo. Dicho genotipo estará, a su vez, formado por cromosomas.
- Los operadores genéticos involucrados en el desarrollo normal del GA acontecen sobre el genotipo.
- La Algorítmica Genética es *robusta*, pues es extrapolable a cualquier tipo de problema, pero *débil* a su vez, pues no está especializada en ninguno en concreto.
- A pesar de que el GA toma por base la observación de los procesos naturales, presenta una sutil pero crucial diferencia con respecto a los mismos: En dichos procesos, el único factor a optimizar es la supervivencia de una especie, maximizando ciertos factores y minimizando otros. No obstante, en el ámbito de los GA solamente se perseguirá la optimización de una única función. En caso de que se busque una optimización que pretenda optimizar diferentes objetivos, deberá adoptarse el esquema de un GA especializado.
- El conjunto de alelos que componen un cromosoma compete con sus homólogos en pos de dilucidar qué elemento del genotipo se asocia con la mejor solución. El entorno algorítmico, representado por el resto de cromosomas, llevará a cabo una presión selectiva sobre la población. De esta forma, únicamente los cromosomas mejor adaptados sobrevivirán, propagando sus genes a su descendencia. Las operaciones genéticas

adscritas a la mutación y el cruzamiento serán un factor clave, pues introducirá el concepto de diversidad genética en el GA.

- Por tanto, a modo de colofón, puede inferirse que un GA pretende, en términos generales, parametrizar el problema, codificar tales parámetros en un cromosoma y aplicar los operadores genéticos, velando siempre por salvaguardar la diversidad genética.

A continuación se presenta un ejemplo de GA que engloba gran parte de las características detalladas a lo largo del presente epígrafe. Para ello, tomando el anteriormente citado videojuego *Quake III Arena* como inspiración, supóngase el siguiente ejemplo:

Un conjunto de ingenieros genéticos pretende desarrollar una criatura absolutamente letal en base a ciertas especies que poseen en su laboratorio, a saber; mariposas, predators, aliens, gatos, terminators, koalas y merluzas. Para ello, desean conocer qué especies deben incluir en sus experimentos en base a un Algoritmo Genético.

Como el lector cinéfilo intuirá, estos científicos estarán interesados en hallar una mezcla de criatura que maximice la presencia de *predators*, *aliens*, y *terminators*, al ser estos los especímenes más letales. (Los vivos colores de una mariposa, la pulcritud de un gato, la pereza de un koala y el afán migratorio de una merluza no parecen aspectos especialmente letales, mientras que sí lo son la ferocidad de los *predators*, la sangre corrosiva de los *aliens* y la cuasi-infinita resistencia de los *terminators*).

Por tanto, y dado que los científicos desconocen la información anterior, se presenta la necesidad de evolucionar ciertas características, en concreto aquellas que permitan desembocar en un ser vivo de exacerbada letalidad. A tenor del supuesto práctico, se sabe con certeza qué características son deseadas, así como aquellas que no lo son tanto en función del ser vivo que deseamos obtener. Por tanto, se persigue *maximizar* una función, o lo que es lo mismo, hallar un conjunto de características de los seres vivos en cuestión ($x_i, x_j, \dots, x_n$) tales que $F(x_i, x_j, \dots, x_n)$ alcance el valor máximo. En otras palabras, el equipo de ingenieros genéticos consta de varios *inputs* para su creación final, y pretende que el *software* establezca cuáles de estos inputs deben estar presentes al final de la evolución (*output*). Los *inputs* representan la lista de cromosomas; el ADN a usar para dar lugar a una criatura absolutamente letal.

Partiendo de las anteriores premisas, se está en posesión de la información suficiente para comenzar a delimitar la Clase perseguida, tomando el lenguaje de programación Java (Version 1.8) como herramienta de implementación. Se aprovecha para informar al lector de que el código fuente de la siguiente

aplicación, al igual que el resto de *software* involucrado en la presente Tesis Doctoral, puede encontrarse en el repositorio GitHub dedicado; <https://github.com/ClemencioMoralesLucas/PhD>. Dicha clase, llamada *UltimateMenaceCreator.java*, que puede encontrarse en la URI <https://github.com/ClemencioMoralesLucas/PhD/blob/master/src/phd/cml/genetic/algorithms/UltimateMenaceCreator.java>, constará de los siguientes atributos:

```
public class UltimateMenaceCreator {  
  
    public static final int BIT_ONE = 1;  
    public static final int BIT_ZERO = 0;  
  
    public static final int GENOTYPE_LENGTH = 10;  
    public static final int CHROMOSOME_LENGTH = 7;  
  
    public static final int NUMBER_OF_ITERATIONS = 100;  
  
    public enum Species {  
        BUTTERFLY, PREDATOR, ALIEN, CAT, TERMINATOR, KOALA, HAKE  
    }  
  
    private int genotypeLength;  
    private int chromosomeLength;  
    private int best;  
    private int numberOfIterations;  
    private boolean[][] chromosome;  
}
```

Por tanto, la clase en cuestión contará con dos flags binarios, una longitud de genotipo (número de individuos de la población), una longitud del cromosoma (número de alelos) y un número de iteraciones a ejecutar el GA. Por otro lado, los especímenes anteriormente citados coexistirán en una estructura tipo enum para una más cómoda y eficiente sintaxis computacional.

Establecidos los atributos requeridos, ¿qué datos debería necesitar un GA genérico para inicializarse? En primer lugar, debe conocer la longitud de la población para conocer de primera mano el número de individuos a evolucionar. Por otro lado, las operaciones genéticas involucradas deben operar a nivel de alelo, por lo que se infiere de forma implícita que el GA en su conjunto debe conocer la longitud de los cromosomas involucrados. Finalmente, el GA debe finalizar al haber transcurrido un determinado número de iteraciones, por lo que este dato es a su vez requerido. Por tanto, el constructor del GA debe recibir la longitud del genotipo (*genotypeLength*), la del cromosoma (*chromosomeLength*) y el número de iteraciones (*numberOfIterations*) que debe ejecutarse. Conocidos tales datos, el GA propuesto debe llevar a cabo la etapa de inicialización. Su codificación, junto con la de sus respectivos *getters* y *setters*, figura seguidamente:

```

public UltimateMenaceCreator(final int genotypeLength, final int
chromosomeLength, final int numberOfIterations) {
    this.setGenotypeLength(genotypeLength);
    this.setChromosomeLength(chromosomeLength);
    this.setNumberOfIterations(numberOfIterations);
    this.initialize();
}

public int getGenotypeLength() {
    return genotypeLength;
}

public void setGenotypeLength(final int genotypeLength) {
    this.genotypeLength = genotypeLength;
}

public int getChromosomeLength() {
    return chromosomeLength;
}

public void setChromosomeLength(final int chromosomeLength) {
    this.chromosomeLength = chromosomeLength;
}

public int getNumberOfIterations() {
    return numberOfIterations;
}

public void setNumberOfIterations(int numberOfIterations) {
    this.numberOfIterations = numberOfIterations;
}

public int getBest() {
    return best;
}

public void setBest(int best) {
    this.best = best;
}

public boolean[][] getChromosome() {
    return chromosome;
}

public void setChromosome(final boolean[][] chromosome) {
    this.chromosome = chromosome;
}

```

Es interesante recordar cuál es el objetivo principal de la función de inicialización de un GA: Tras establecer la longitud de los cromosomas, que en el escenario que nos ocupa vendrá representada por una matriz, estos serán inicializados siguiendo un esquema de corte aleatorio [Ting, 2005]. En el *software* en cuestión, se generarán *bits* aleatoriamente con el fin de lograr dicho objetivo:

```

private void initialize(){
    //GA Population
    this.setChromosome(new
boolean[this.getGenotypeLength()][this.getChromosomeLength()]);

    //Random Genotype Initialization

```

```

    for (int i = 0; i < this.getGenotypeLength(); i++) {
        for (int j = 0; j < this.getChromosomeLength(); j++) {
            chromosome[i][j] = getRandomBoolean();
        }
    }
}

```

Adviértase que en el fragmento de código fuente superior el valor cromosómico se representa mediante un valor de tipo booleano, pues únicamente se está en necesidad de saber qué especie debe estar presente en la creación final. Por consiguiente, el valor lógico a nivel alto (BIT_ONE) indicará que la especie está presente, ocurriendo lo contrario para el valor lógico a nivel bajo (BIT_ZERO).

De forma paralela, en cuanto a la parte más puramente algorítmica se refiere, resta por llevar a cabo la operación, quizá más crítica, en el ámbito de un GA: Evolucionar el *genotipo* hasta alcanzar el *fenotipo* deseado. En esta importante misión, la calidad de cada cromosoma se medirá, como se ha descrito previamente a lo largo del presente epígrafe, mediante la función de aptitud o *fitness*. Como ya se ha especificado, tal función "premiará" la presencia de *predators*, *aliens* y *terminators* en la misma cuantía, mientras que calificará negativamente a aquellos alelos que representen los genes de mariposas, gatos, koalas o merluzas. Por tanto, si una especie en concreto debe estar presente en la creación final, esta recibirá un valor positivo o alto, mientras que si ocurre lo contrario la especie recibirá un valor negativo o bajo en pos de la optimización de la función. Suponiendo que cada cromosoma viene representado por un *array* de alelos en los que las posiciones siguen el mismo orden que el especificado en el *enum*, la función de fitness presenta la siguiente codificación:

```

private static int fitnessFunction(final boolean[] chromosome) {
    final int[] intChromosome = parseToIntArray(chromosome);

    //Species available (We'd like to mix 1, 2 and 4 mainly):
    // 0          1          2          3          4          5          6
    //BUTTERFLY, PREDATOR, ALIEN, CAT, TERMINATOR, KOALA, HAKE

    return (-intChromosome [0] + intChromosome [1] + intChromosome [2]
            -intChromosome [3] + intChromosome [4] - intChromosome [5]
            -intChromosome [6] );
}

```

Por otro lado, de forma paralela en tan crítica función en el GA, el mecanismo de evolución a implementar (en este caso basado en el Principio Algorítmico Genético del *Elitismo*) deberá iterar sobre el genotipo calculando las funciones de aptitud de cada cromosoma y almacenando el mejor valor, que, eventualmente, se corresponderá con el fenotipo. Posteriormente, se deben aplicar los operadores genéticos de la reproducción (*crossover*) y mutación (*mutation*). El primero de ellos, tal y como se ha explicado en los primeros compases del presente epígrafe, intercambiará aleatoriamente alelos entre los cromosomas

progenitores, mientras que el segundo variará, de nuevo aleatoriamente, los alelos pertenecientes al cromosoma con una ínfima probabilidad, en este caso del 4%:

```
public void evolve(final boolean verboseMode){
    best = 0; //Index of the best element among the population
    for (int g = 0; g < this.getNumberOfIterations(); g++) {
        if(verboseMode){
            System.out.println("Generation #" + g);
        }

        //Genotype Evaluation
        for (int i = 1; i < this.getGenotypeLength(); i++) {
            if (fitnessFunction(chromosome[best]) <
fitnessFunction(chromosome[i])) {
                best = i;
            }
        }
        //Genetic operators application
        for (int i = 0; i < this.getGenotypeLength(); i++) {
            //Crossover that avoids self-reproduction
            if (i != best) {
                for (int j = 0; j < this.getChromosomeLength(); j++){
                    if (getRandomBoolean()) {
                        chromosome[i][j] = chromosome[best][j];
                    } else {
                        chromosome[i][j] = chromosome[i][j];
                    }
                }
                //Mutation (lower probability)
                if (new Random().nextInt(100) % 100 < 4) {
                    chromosome[i][j] = getRandomBoolean();
                }
            }
        }
        if(verboseMode){
            printCurrentPhenotype(chromosome[best]);
        }
    }
}
```

Es reseñable que, con el fin de posibilitar la existencia de un fichero de tipo *log* y hacer más cómodo el proceso de *tracing* del *software*, la función cuenta con el flag booleano `verboseMode`, que en caso de obtener el valor `true` imprimirá el estado del fenotipo actual vía consola.

Por tanto, se está en disposición de esbozar el método `main` de la clase `UltimateMenaceCreator.java` que únicamente deberá instanciar el objeto homónimo y atacar a su método `evolve`, para posteriormente imprimir los resultados mediante los métodos de utilidad implementados:

```

public static void main(String [] args) {
    UltimateMenaceCreator ultimateMenaceCreator =
        new UltimateMenaceCreator(GENOTYPE_LENGTH,
        CHROMOSOME_LENGTH, NUMBER_OF_ITERATIONS);

    final boolean verboseMode = true;
    ultimateMenaceCreator.evolve(verboseMode);
    ultimateMenaceCreator.printPhenotype();
}

```

Como colofón, se presenta el conjunto de métodos de utilidad utilizados para generar bits de manera aleatoria e imprimir el conjunto de soluciones:

```

private static int[] parseToIntArray(final boolean[] booleanArray) {
    final int[] intArray = new int[booleanArray.length];
    for (int i = 0; i < booleanArray.length; i++) {
        intArray[i] = booleanArray[i] ? BIT_ONE : BIT_ZERO;
    }
    return intArray;
}

private static boolean getRandomBoolean() {
    return Math.random() < 0.5;
}

public void printPhenotype(){
    System.out.println();
    System.out.println("***** PHENOTYPE FOUND *****");
    System.out.println("Best fitness: " +
    fitnessFunction(chromosome[best]));
    printBestPhenotype(chromosome[best]);
    System.out.println();
}

private static void printCurrentPhenotype(boolean[] chromosome) {
    System.out.println("Fitness: " + fitnessFunction(chromosome));
    System.out.println(Arrays.toString(chromosome));
    System.out.println("-----");
}

private static void printBestPhenotype(final boolean[] chromosome){
    String result = "Ultimate menace formed by the mixture of:\n";
    String discarded = "Discarded specimens:\n";
    for(int i = 0; i < chromosome.length; i++) {
        if(chromosome[i]){
            result += Species.values()[i] + ",";
        } else {
            discarded += Species.values()[i] + ",";
        }
    }
    result = result.replaceAll(",$", "") + "];";
    discarded = discarded.replaceAll(",$", "") + "];";

    System.out.println(result);
    System.out.println();
    System.out.println(discarded);
}

```


Tras ejecutar el Algoritmo Genético anteriormente planteado, por ejemplo en el IDE (*Integrated Development Environment*; Entorno de Desarrollo Integrado) *IntelliJ IDEA* 16.1.4, se obtienen los siguientes resultados, que demuestran empíricamente la rectitud del GA desarrollado, como se argumentará posteriormente:

```
Run UltimateMenaceCreator
-----
Generation #98
Fitness: 3
[false, true, true, false, true, false, false]
-----
Generation #99
Fitness: 3
[false, true, true, false, true, false, false]
-----
***** PHENOTYPE FOUND *****
Best fitness: 3
Ultimate menace formed by the mixture of:
[PREDATOR,ALIEN,TERMINATOR]

Discarded specimens:
[BUTTERFLY,CAT,KOALA,HAKE]

Process finished with exit code 0
Compilation completed successfully in 27s 447ms (today 16:45)
```

Figura XXVII: Resultado de la ejecución del GA representado en *UltimateMenaceCreator.java*

A tenor de la figura anterior, se puede comprobar cómo las cien generaciones son suficientes para encontrar el genotipo perseguido, que presenta un valor Best fitness de tres, lo que representa que, tal y como la intuición apuntaba, dicho número de especies han sido elegidas (Ultimate menace formed by the mixture of: Predator, Alien, Terminator), mientras que el resto han sido descartadas (Discarded specimens: Butterfly, Cat, Koala, Hake). Se destaca que la Técnica del *Elitismo* utilizada en el presente GA, pretende seleccionar al mejor individuo de la población y hacerlo reproducirse con el resto de especímenes. Tal y como se especifica anteriormente en el presente epígrafe, la suerte de operación genética en la que dos organismos progenitores dan lugar a una nueva generación se denomina *Cruzamiento* (*Crossing Over*). Aunque esta característica mimetiza fielmente el comportamiento natural de los seres vivos, la Mutación introducida dota a GA de un carácter más realista aún, si cabe, pues simboliza el efecto del ecosistema en la población, aumentando la diversidad de la misma. Llegado este punto se aprovecha para referenciar al

lector que desee visualizar el código fuente en su completitud al repositorio *GitHub* previamente especificado.

Una vez introducido al lector a la programación de los GA y concluyendo el presente epígrafe, se presentan una serie de recomendaciones que no se deberían perder nunca de vista a la hora de lidiar con el diseño algorítmico de un GA. Tales conclusiones se infieren del importante artículo publicado por Goldberg, insigne pupilo de Holland, en la Conferencia Internacional de los Algoritmos Genéticos en 1989 (*International Conference on Genetic Algorithms*; ICGA en lo sucesivo):

- *Watch out for the frontal assault* (Cuidado con el asalto frontal): En ciertas ocasiones, se puede caer en la tentación de plantear el problema de pérdida de diversidad genética en la población cromosómica. En este caso, existen dos vertientes para solucionar el problema:
 - Convertir el GA en un Algoritmo de Búsqueda Aleatoria (ABA): Esta transición puede llevarse a cabo de forma sencilla en base a aumentar el ritmo de mutación.
 - Variar la función de aptitud: Mediante mecanismos de *sharing*, la función de *fitness* o aptitud de un individuo puede dividirse por el número de cromosomas equivalentes al mismo. Adviértase que esta metodología mimetiza de manera más fiel el comportamiento de la naturaleza, pues cada individuo, incluso a pesar de que presente una calidad elevadamente atípica, debe compartir recursos con los homólogos que resuelvan el problema.
- *Let nature be your guide* (Permite que la naturaleza sea tu guía): Puesto que un gran porcentaje de los problemas que se pretenden solucionar mediante algorítmica genética son de naturaleza no lineal, es más efectivo simular el comportamiento de la naturaleza, aunque de forma intuitiva pueda no parecer una acción recomendable. El propio Goldberg establece que *"if we want to develop no-linear systems that search and learn, it's better to start, at least, imitating working systems. And these systems can be found within Nature."* (Si queremos desarrollar sistemas no lineales que busquen y aprendan, es mejor comenzar, al menos, imitando sistemas que funcionan. Y estos sistemas pueden hallarse en la naturaleza.)
- *Do not trust the central authority* (No confíes en la autoridad central): Dado que los procesos naturales están impregnados de un carácter distribuido, debe minimizarse el cardinal de operadores que disfruten de una visión global del genotipo. Por definición, la adopción del anteriormente citado esquema constituye una autopista a la optimización del GA mediante la

programación paralela, pues, como se cita en áreas anteriores del presente epígrafe, cada cromosoma puede estar hospedado en un nodo, posibilitándose su tratamiento a nivel de *thread*. En caso de que la comparación de la función de aptitud entre los cromosomas sea necesaria, se debe delimitar un *vecindario* reducido de cromosomas, entre los que se llevará a cabo la comparación, huyendo de llevar a cabo dicha comparación entre el genotipo en su conjunto.

- *Respect the schema riddle* (Respeto a la criba de esquemas): Esta importante recomendación, en la que ya se ha enfatizado en el presente epígrafe, insta a utilizar alfabetos de cardinalidad ínfima para representar los alelos que componen los cromosomas. Idealmente, esta representación será binaria, pues el número de caracteres del alfabeto utilizado será inversamente proporcional al rendimiento del GA.

2.1.3 Ant Colony Optimization

Parafraseando al autor de la presente vertiente algorítmica adscrita a la *Computación Natural*, Marco Dorigo, así como a su ayudante Gianni Di Caro, en su obra *The Ant Colony Optimization Meta-Heuristic* [Dorigo, 1992], se denomina Algoritmo de la Colonia de Hormigas o *Ant Colony Optimization* a aquel sistema de corte multi-agente en el que el comportamiento de cada agente, denominado *hormiga artificial* u *hormiga*, está inspirado en el comportamiento real en cuanto a la búsqueda de sustento de dichos formícidos.

Es digno de mención que el algoritmo que bautiza el presente epígrafe pertenece al conjunto de paradigmas, técnicas y metodologías computacionales denominado *Swarm Computing* (traducido como Computación o Inteligencia de Enjambre). Dicha técnica, adscrita al ámbito de la Inteligencia Artificial, tiene como objetivo el estudio del comportamiento colectivo de todos aquellos sistemas auto-organizados y descentralizados independientemente de su origen, que puede ser de origen natural o artificial. El concepto *Swarm Computing*, surgido en 1989 de la mano de los científicos de la computación Wang Jing y Gerardo Beni como fruto de la investigación en el contexto de los RMS (*Mobile Robotic System*; Sistema Robótico Móvil), se asocia con aquellos sistemas formados por un conjunto de agentes simples que interactúan entre sí y través de su *entorno* de forma concurrente. Una de las principales características de la Inteligencia de Enjambre se corresponde con el comportamiento de los agentes involucrados en sus procesos, que responden, siguiendo un esquema clásico, en base a sencillas directrices [Bonabeau et al., 1999]. Es digno de mención que, a pesar de que el comportamiento de los agentes no suele ser complejo en exceso, el sumatorio de infinitesimales provoca que el sistema *per se* sí lo sea; la gran cantidad de agentes que los algoritmos adscritos a la Inteligencia de Enjambre involucra, provoca que el comportamiento global del sistema sea especialmente complejo. Baste citar, a modo de ejemplo, el comportamiento seguido por seres gregarios como el alineamiento, casi perfecto desde el punto de vista geométrico, de las aves en vuelo, o, como se demostrará a lo largo del presente epígrafe, la distribución estratégica de las colonias de hormigas. Con el fin de demostrar empíricamente las aseveraciones anteriores, se presenta al lector una curiosa figura que ilustra el vuelo de una bandada de *Columba Livia* o paloma común:

Figura XXVIII: Figura geométrica compuesta por una bandada de *Columba Livia* en base a la Inteligencia de Enjambre

Englobando las características anteriormente descritas, pueden inferirse las premisas principales de un Sistema de Enjambre:

- Algorítmica distribuida, carente de fuente de control central u origen específicamente señalado de información.
- Comunicación especialmente limitada entre los agentes que componen el sistema.
- Carencia de modelo explícito del ecosistema o entorno de los agentes involucrados.
- Necesidad, por parte de los agentes, de una excelente habilidad de captación del entorno que les rodea.
- Habilidad de las entidades involucradas en el sistema de reaccionar, idealmente con celeridad, a los cambios en el ecosistema aglutinante.

Introducido el concepto de *Swarm Computing*, y retomando la vertiente algorítmica que da nombre al presente epígrafe, se destaca que los Algoritmos de Colonia de Hormigas, ACO en lo sucesivo, toman su inspiración en el Experimento de Goss utilizando una colonia de *Iridomyrmex humilis*, comúnmente denominadas hormigas argentinas [Dorigo, 1992]. Dicho experimento parte de la

premisa de otorgar a los formícidos el libre acceso a una fuente de alimento en un área comunicada con el hormiguero. Tal alimento se encuentra unido al hormiguero por dos posibles rutas, en las que se verifica que una es más larga que la otra, tal y como ejemplifica la siguiente figura:

Figura XXIX: Ejemplo esquemático ilustrativo del Experimento de Goss

El experimento en cuestión, tras permitir a las hormigas desplazarse en base al libre albedrío, permite observar cómo, tras una fase transitoria en la que dichos artrópodos realizan una suerte de análisis del entorno, la mayoría de los mismos pasan a utilizar siempre el camino más corto hacia la fuente de sustento. Por otro lado, es reseñable que, a tenor de dicho experimento, se extrae que la probabilidad de selección de la ruta más corta es directamente proporcional a la diferencia de tamaño entre ambas rutas. (En otras palabras, si la ruta más larga tiene una longitud significativamente superior a su homóloga, la probabilidad de que los formícidos opten por la ruta más larga es ínfima.)

Una crucial cuestión que puede asaltar de forma natural al lector tras estudiar los resultados del Experimento de Goss, es la asociada a la comunicación entre los formícidos, y es que, ¿cómo es posible que dichos artrópodos puedan comunicarse de forma tan óptima? La respuesta a esta pregunta se basa en el concepto conocido como *estigmergia*:

Se denomina *estigmergia* a aquel protocolo de colaboración, a través del medio físico, en el que los distintos componentes colaboran en base a acumulaciones de objetos o magnitudes en su entorno, tales como feromonas o humedad, respectivamente.

Tal y como se detallará a continuación, es, precisamente, el concepto de feromona el que toma un papel vital en la comunicación entre los formícidos; mientras las hormigas se desplazan desde el hormiguero hacia el alimento y viceversa, depositan una sustancia de carácter químico, llamada *feromona*, en el suelo. Es digno de mención que las *feromonas* son sustancias químicas secretadas por un gran número de seres vivos, cuya finalidad pretende provocar

específicas reacciones y comportamientos en sus homólogos de la misma especie [Dorigo et al., 1999].

Ahora bien, ¿cómo influyen las anteriormente citadas feromonas en la toma de decisiones de los artrópodos que nos ocupan? La respuesta es, ciertamente, intuitiva, demostrando una vez más la elegancia que impregna los sistemas diseñados por la naturaleza; cuando una hormiga, que desconoce dónde se encuentra el alimento, llega a una bifurcación, esta realiza una decisión de origen probabilístico basado en la cantidad de feromonas presente en la entrada a cada camino. Tal característica presenta un carácter auto catalítico, pues la elección de un determinado camino por parte de una hormiga incrementará la probabilidad de que sus homólogos tomen la misma ruta.

Tal y como el atento lector intuirá, esta suerte de algorítmica natural necesita una fase de *inicialización* que busca la impregnación de feromonas en las rutas existentes. En el instante inicial del Experimento de Goss, no existe ningún tipo de feromona en ningún camino hacia la fuente de alimento. Por tal motivo, los formícidos elegirán cualquier camino existente entre el hormiguero y la fuente de alimento en base a, en términos estadísticos, una Distribución Uniforme. En otras palabras, la probabilidad de que un camino sea elegido por una hormiga es la misma que la de sus homólogos. Posteriormente, los formícidos comenzarán a explorar su entorno, liberando feromonas a su paso. Eventualmente, los artrópodos hallarán la fuente de alimento, por lo que comenzarán el camino de retorno, cargados de alimento, hacia el hormiguero, liberando de nuevo feromonas a su paso. Transcurridos unos minutos, el camino más corto habrá sido transitado por un mayor número de hormigas, por lo que la cantidad de feromonas en el mismo será mayor, factor que provocará que sea transitado por un cada vez mayor número de formícidos que, eventualmente, serán todas las hormigas disponibles en la búsqueda.

Ahora bien, ¿qué ocurre con las feromonas depositadas en el camino más largo? ¿No provocaría su presencia un desvío de los artrópodos que impactaría negativamente en el rendimiento del algoritmo? La respuesta a esta pregunta está directamente relacionada con el factor de *volatilidad* presente en las feromonas; Cuando tal sustancia química es depositada en una ruta, la feromona se *evaporará* transcurrido un cierto tiempo. Por esta razón, las feromonas requieren de una suerte de sistema de feedback positivo para subsistir. Cuando las hormigas, en su camino de retorno al hormiguero, alcancen el punto de decisión o bifurcación, observarán la presencia de feromonas en el camino más corto. Dicho rastro, que será liberado durante el viaje de ida hacia la fuente de alimento, indicará la ruta hacia el camino más corto. Por tanto, a pesar de que existirán feromonas en el camino más largo en las primeras etapas del Experimento de Goss, la no insistencia de hormigas en tal camino provocará la evaporación de las feromonas. De este modo, el desvío de los artrópodos

únicamente ocurre al inicio del algoritmo, siendo prácticamente despreciable en su conjunto.

Por tanto, adviértase que, conforme el número de iteraciones en el algoritmo crece, la feromona existente en el camino más corto se deposita con una mayor frecuencia que en el más largo, haciendo la ruta más corta más apetecible para los artrópodos.

Una vez otorgada una introductoria pero exhaustiva descripción sobre el Paradigma *Ant Colony Optimization*, se está en disposición de describir, de manera formal, el Algoritmo de la Colonia de Hormigas *per se*. Para ello, es recomendable recordar al lector la útil estructura matemática conocida como *Grafo*; Se denomina *Grafo* a un a toda dupla G , donde:

$$G = (V, A)$$

Donde V representa al conjunto de *Vértices* y A al conjunto de *Aristas* que unen los elementos pertenecientes al conjunto V . Es digno de mención que un *Grafo* puede ser ponderado, siempre y cuando sus aristas tengan asociado un valor numérico, y/o dirigido, cuando existe un sentido en la circulación de la arista existente entre dos vértices. A continuación se presenta un *Grafo* no dirigido ponderado:

Figura XXX: Ejemplificación de la entidad matemática: *Grafo no dirigido ponderado*

En el Algoritmo de la Colonia de Hormigas, el *plano* que representa la ubicación

del hormiguero, los caminos existentes a disposición de los formícidos y la fuente de alimento se corresponderá con un *Grafo*. La tarea principal de cada artrópodo virtual es la de encontrar el camino más corto entre dos vértices del *Grafo* en el que el itinerario está representado. De forma previa a la inmersión en los aspectos más íntimos del Algoritmo de la Colonia de Hormigas se presenta la siguiente figura, que pretende aunar los conceptos anteriormente detallados recordando al lector la correlación entre las entidades extraídas del mundo real y su extrapolación computacional:

Mundo Natural	Mundo Computacional
Ecosistema/ Entorno	Grafo
Hormiguero y fuente de alimento	Vértices adscritos al grafo; nodo inicial y final
Hormiga	Agente
Feromona	Rastro/ Feromona artificial

Figura XXXI: Correlación entre los conceptos del mundo natural y el computacional en el ámbito del ACO

Recapitulando, suponiéndose un escenario en el que el *grafo* sea *conexo*, o lo que es lo mismo, si para cualquier par de *vértices* u y v en el *grafo*, existe al menos una *trayectoria* (una sucesión de vértices adyacentes que no repita vértices) de u a v , con $n=|N|$ nodos, el algoritmo en cuestión puede ser utilizado con el fin de hallar la solución adscrita al camino más corto entre el hormiguero y la fuente de sustento. De esta forma, la solución es un camino desde un *vértice* A hasta su homólogo B , donde la longitud del camino viene dada por el número de saltos entre ambos puntos. Es digno de mención que la longitud puede ser cuantificada de una manera más efectiva si el *grafo* es *ponderado*, puesto que el *coste* del camino vendrá dado por el sumatorio de los valores asociados a cada *arista* por la que se transita entre los vértices A y B . A continuación se ilustra un escenario explicativo en el que un *formícido* encuentra el camino más corto desde el hormiguero (vértice superior izquierdo) hasta la fuente de sustento (vértice inferior derecho):

Figura XXII: Representación esquemática del Algoritmo de la Colonia de Hormigas

Para cada arista (i, j) del grafo en cuestión, se asocia una variable τ_{ij} denominada rastro de feromonas artificial. Tal rastro es interpretado y escrito a su vez por el conjunto de artrópodos involucrados en el algoritmo, siendo la intensidad de cada feromona proporcional a su utilidad. Consiguientemente, una feromona ubicada en el camino más corto entre el hormiguero y el alimento presentará una intensidad mayor que su homóloga presente en uno de los caminos más largos entre dicho origen y destino. En el Algoritmo de la Colonia de Hormigas, cada formícido aplica una política de resolución paso a paso, que, eventualmente, finalizará al hallarse la ruta más corta hacia la fuente de alimento. En cada vértice del grafo que representa el itinerario, se dispondrá de la información asociada a las aristas entrantes y salientes. Dicha información será utilizada para determinar, de manera estocástica, cuál es el siguiente nodo al que el formícido debe acudir.

Cabe destacar que la directriz por la que cada hormiga k ubicada en el nodo i utiliza el rastro de feromonas τ_{ij} para hallar la probabilidad con la que debe elegir un nodo j perteneciente a N_i , así como el siguiente nodo al que desplazarse, donde N_i constituye el conjunto de nodos adyacentes a i , viene dado por:

$$p_{ij}^k \propto \tau_{ij} \text{ si } j \in N_i \quad 0 \text{ si } j \notin N_i$$

Adviértase que, típicamente y con fines de optimización, es frecuente inicializar el Algoritmo de la Colonia de Hormigas con una ínfima cantidad de feromonas depositadas por defecto en todas las aristas. En el instante inicial, se verifica que la cantidad de feromonas existente para cualquier arista es idéntica a la presente en cualquier homóloga.

Tal y como se ha indicado a lo largo del presente epígrafe, a medida que los formícidos delimitan una solución para el algoritmo, estos depositan cierta cantidad de feromonas en las aristas por las que transitan. En el Algoritmo de Colonia de Hormigas clásico, la cantidad de feromonas, $\Delta\tau$, es constante. Suponiéndose una hormiga que, en un tiempo t , transita desde el vértice i hacia el nodo j , el valor de la feromona, τ_{ij} , varía en base a la siguiente fórmula [Dorigo et al., 2004]:

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau$$

Por tanto, a tenor de la fórmula anteriormente especificada, que simula la cantidad real de feromonas depositada en la arista (i,j) , un formícido que utiliza la ruta $i-j$, incrementa la probabilidad de que sus homólogos utilicen la misma ruta en el futuro. Tal y como ocurre con las hormigas en el mundo real, la autocatálisis y la diferencia de longitud entre caminos actuarán como un punto a favor

del surgimiento de caminos cortos, óptimos por ende. Es digno de mención que, como se ha indicado en anteriores secciones y a fin de evitar una convergencia súbita de los formícidos hacia un óptimo local y no global, se añade el algoritmo un mecanismo de exploración de respaldo; de forma análoga a los rastros de feromonas reales, sus homólogos virtuales se evaporan. Por consiguiente, la intensidad de cualquier feromona decrece en función del tiempo [Dorigo et al., 1997], favoreciendo la exploración de diferentes aristas durante el proceso de búsqueda. Es reseñable que el proceso de evaporación se lleva a cabo mediante el decremento de los rastros de feromonas siguiendo una Distribución Exponencial, donde se aplica la siguiente fórmula en cada iteración del Algoritmo de la Colonia de Hormigas:

$$\tau \leftarrow (1 - \rho) \tau, \quad \rho \in (0, 1]$$

Llegado este punto, es de vital importancia reseñar que si la complejidad del grafo es elevada, (p. ej., existe más de un camino entre el hormiguero y la fuente de sustento), el rendimiento del algoritmo tiende a desestabilizarse, por lo que el ajuste del valor otorgado a los parámetros de entrada al algoritmo es crítico, entendiéndose por valores de entrada la ponderación de cada arista, por citar un ejemplo.

Adviértase que, como prácticamente todo algoritmo que se precie, el Algoritmo de Colonia de Hormigas clásico presenta ciertas carencias. La mejora de tales carencias puede ejecutarse mediante numerosas aproximaciones como, por ejemplo, la cuantificación de las feromonas: En lugar de que cada formícido deposite siempre la misma cantidad de feromonas en las rutas por las que transita, estas feromonas puede variar en cuanto a su cuantía, siendo la cantidad de feromonas directamente proporcional a la solución que está siendo generada, en un instante t , por dicho artrópodo. Por otro lado, puede almacenarse cierta información, de corte heurístico, en los vértices. De este modo, se posibilita la benigna opción de que las hormigas puedan estudiar la necesidad de transitar por el vértice *a priori*.

De forma paralela, una perniciosa característica de las hormigas descritas en los párrafos anteriores es la *carencia de memoria*. Por tal motivo, el Algoritmo de la Colonia de Hormigas clásico puede ser únicamente aplicado a la búsqueda del camino más corto en un itinerario sin restricciones adicionales. En caso de que se deseen hallar las posibles soluciones adscritas a problemas de más compleja índole, como el camino *Hamiltoniano* más corto en un grafo, los formícidos deben disponer de cierta capacidad para memorizar la información pertinente.

Englobando los razonamientos esgrimidos a lo largo del presente epígrafe, la

lógica atómica del Algoritmo de la Colonia de Hormigas puede delimitarse como se expresa a continuación:

Figura XXIII: Representación gráfica del Algoritmo de la Colonia de Hormigas

1. Tomando como el hormiguero (N ; *Nest*) como origen de coordenadas, una hormiga parte hacia la búsqueda de sustento (F ; *Food*), recorriendo, en el peor de los casos, todos los caminos disponibles en el escenario (a , b). Una vez encontrada la fuente de alimento, el formícido inicia el viaje de regreso al hormiguero, desplegando feromonas a su paso.
2. El resto de artrópodos pertenecientes al hormiguero se lanzarán a la búsqueda de sustento. A pesar de que en las primeras etapas del algoritmo existirán feromonas en rutas no óptimas (*inicialización*), la convergencia obtenida con el paso del tiempo provocará que la ruta más corta sea la más frecuentada y, por ende, la ruta que mayor concentración de feromonas presenta, pues los formícidos siempre tenderán a elegir la ruta de menor longitud hacia el alimento.
3. Los artrópodos se decantan por el itinerario más corto hacia la fuente de sustento, mientras que las feromonas presentes en el resto de rutas empiezan a evaporarse debido a la carencia de *feedback*, entendiendo como tal el paso de artrópodos por el mismo.

Tal y como puede extraerse a partir del estudio de los postulados esgrimidos hasta el momento, los formícidos hacen de su entorno el más efectivo medio de comunicación. En base al algoritmo, los artrópodos definen una suerte de comunicación *implícita*, en la que el despliegue de hormonas en la ruta más corta supone el acuerdo tácito a fin de hallar la ruta óptima hacia la fuente de alimento. Adviértase que las feromonas se *intercambian* únicamente en el ámbito más

puramente local, delimitando un comportamiento propio de la *estigmergia*, que típicamente se presenta en los seres *gregarios*. Queda evidenciado, por tanto, cómo la metodología a seguir en pos de la resolución de un problema complejo no siempre ha de tener el mismo grado de dificultad que la cuestión que pretende resolver. Tan benigna característica, en este caso, se obtiene gracias a un doble mecanismo de *feedback*:

- *Feedback positivo*: Provocado por el despliegue de feromonas por parte de los artrópodos en una ruta, motiva la atracción de las hormigas hacia dicha ruta.
- *Feedback negativo*: Cuando las hormigas dejan de frecuentar una ruta, las feromonas depositadas en la misma se evaporarán, por lo que los formícidos dejarán de frecuentarla.

Adviértase que, desde el punto de vista puramente teórico, si la cantidad de feromonas durante la ejecución del algoritmo fue la misma durante el tiempo de ejecución, ninguna ruta fue "realmente" elegida. No obstante, gracias al sistema de *feedback* anteriormente descrito, la más mínima variación positiva de feromonas en una arista del grafo se traducirá en la elección de dicha ruta por parte de los formícidos. Por tanto, el algoritmo partirá desde un estado de inestabilidad en el que todas las aristas disfrutaran de la misma cantidad de cromosomas, hasta, finalmente, un estado totalmente estable donde el itinerario más corto hacia la fuente de sustento está delimitado por las aristas con un mayor número de feromonas.

Englobando los conceptos anteriormente descritos, es reseñable que, a modo de colofón, la base del algoritmo implica el desplazamiento de un conjunto de formícidos a través de los distintos sub-escenarios del problema influenciado por una política de decisión bilateral, basada en las *rutas disponibles* y la atracción causada por las *feromonas*. Una vez un artrópodo delimita una solución, sus homólogos la evaluarán y modificarán a tenor de las circunstancias propias de cada escenario, por lo que el algoritmo está impregnado de un crítico sistema de valoración constante. Adviértase que tan exigente protocolo, unido a la característica de *evaporación* de las feromonas, desemboca en una disminución drástica de la probabilidad de hallar un óptimo local y no global.

A pesar de la complejidad aparente en ciertos aspectos del Algoritmo de la Colonia de Hormigas, como se apreciará a continuación su *pseudocódigo* es ciertamente conciso [Dorigo, 2007]:

```

public void AntColonyOptimization(){

    while(terminationConditionNotPresent()){

        searchForSolutions();

        backgroundActions();

        updatePheromones();

    }

}

```

Con respecto al pseudocódigo anteriormente presentado, destacan dos secciones algorítmicas de carácter no trivial:

- *Procedimiento de actualización de feromonas*: Una vez los formícidos han hallado una posible solución, todos los rastros de feromonas involucrados en el algoritmo son actualizados mediante la fórmula:

$$\tau_{xy} \leftarrow (1 - \rho) \tau_{xy} + \sum_k \Delta\tau_{xy}^k$$

Donde τ_{xy} se corresponde con la cantidad de feromonas desplegadas para el estado de transición xy , ρ se corresponde con el coeficiente de evaporación de feromonas y $\Delta\tau_{xy}^k$ la cantidad de tal sustancia química desplegada por la hormiga k -ésima. Cabe destacar que tal valor puede hallarse, a su vez, mediante la fórmula:

$$\Delta\tau_{xy}^k \begin{cases} \frac{Q}{L_k} & \text{si el formícido } k \text{ usa la curva } xy \text{ en su ruta} \\ 0 & \text{en otro caso} \end{cases}$$

Donde se verifica que L_k se corresponde con la cuantía de la ruta de la k -ésima hormiga y Q es constante.

- *Procedimiento de selección de aristas*: Partiendo de la base de que cada formícido adscrito al algoritmo se corresponde con un *agente computacional*, el algoritmo parte de la construcción iterativa de una solución para el escenario que se pretende resolver. Para cada *paso* o iteración del Algoritmo de la Colonia de Hormigas, cada artrópodo transita desde un estado x hacia un estado y , computando durante el proceso un conjunto $A_k(x)$ de expresiones propias de su estado actual en cada iteración mediante un enfoque heurístico. A su vez, para cada hormiga k , es reseñable que la probabilidad p_{xy}^k de moverse desde el estado x hacia y viene dada por el influjo de la dupla n_{xy} del movimiento de los formícidos, así como de la probabilidad acorde a la conveniencia del movimiento y la cantidad de feromonas T_{xy} del mismo. De esta forma, se posibilita la

computación de un *feedback* que indicará el grado de acierto tras la realización de dicho movimiento. Por otro lado, es digno de mención que el nivel del rastro es actualizado una vez todo formícido adscrito al algoritmo ha hallado su solución parcial, por lo que dicho rastro será incrementado o decrementado en función de la benignidad de la solución hallada. En cuanto al aspecto más puramente probabilístico [Beni et al., 1989] se refiere, el formícido k -ésimo transita desde el estado x hacia y en base a la fórmula:

$$p_{xy}^k = \frac{(\tau_{xy}^\alpha)(\eta_{xy}^\beta)}{\sum (\tau_{xy}^\alpha)(\eta_{xy}^\beta)}$$

Donde se verifica que T_{xy} es la cantidad de feromonas desplegadas en el proceso de transición desde x a y , $0 \leq \alpha$ a su vez representa el parámetro encargado de controlar el grado de influencia de T_{xy} , N_{xy} es la conveniencia asociada al estado de transición x hacia y y $\beta \geq 1$ es el parámetro de entrada encargado de controlar la influencia ejercida por el valor N_{xy} . Es reseñable que xy representa un conocimiento previo, que suele ser d_{xy}^{-1} , donde d es la distancia en cuestión.

Habiéndose descrito la capa matemática adscrita al ACO, procede hacer especial énfasis en ciertos aspectos referidos a dicho algoritmo. En primer lugar es reseñable cómo el ACO realiza una evaluación *implícita* y *explícita* a su vez. Tal comportamiento es provocado por la retroalimentación otorgada por los artrópodos para con sus homólogos en las posteriores iteraciones del algoritmo, mediante una forma explícita consistente en la evaluación de la solución generada para decidir qué cantidad de feromonas debe ser depositada, a la vez que mediante de manera implícita explotando la diferencia de longitud entre rutas tal y como ocurre con los formícidos del mundo real. En otras palabras, este comportamiento se traduce en que el hecho de que una hormiga seleccione un camino corto impactará en los siguientes formícidos, puesto que dicho camino será impregnado de feromonas. Cabe destacar que esta técnica es efectiva en la búsqueda de soluciones óptimas en problemas de enrutamiento de redes, presentando una más que aceptable convergencia en el hallazgo de la solución perseguida. Tan benigno efecto tiene su explicación en la naturaleza puramente distribuida de los nodos de una red (al igual que ocurre con los vértices en el grafo adscrito al ACO), que permite la explotación de la técnica de forma natural, sin incurrir en elevados costes computacionales. El carácter asíncrono de las redes de computadores ayuda en gran medida a este factor, puesto que desliga la problemática de tediosos requisitos temporales.

En contraposición, esta ventaja no está presente en los problemas de

optimización de corte combinatorio, donde la forma más recomendable de implementar una variación del ACO parte de la necesidad de implementar una suerte de hormigas síncronas, es decir, formícidos que añaden nuevos elementos a la solución en construcción en base a un esquema coordinado, síncrono por ende. Aunque esta variación es eficaz, pues resuelve el problema propuesto, no es en absoluto eficiente, o lo que es lo mismo, no resuelve el problema de la mejor manera posible. Esto es debido a que se producen importantes pérdidas en el rendimiento del sistema al existir una necesidad imperiosa de sincronización, que provocará que existan casos en los que determinados formícidos permanezcan ociosos esperando las soluciones de sus homólogos que eligieron la ruta de mayor longitud.

Por otro lado, y como se citó anteriormente, después de que un formícido halle una solución el coste de la misma es usado para computar la cantidad de feromonas que el artrópodo debe depositar en las aristas del grafo que representa el escenario. Por ejemplo, la hormiga puede desplegar una cantidad de feromonas inversamente proporcional al coste de la solución generada, lo que provocaría, en otras palabras, que una gran cantidad de feromonas serían depositadas en una ruta de coste ínfimo. Es reseñable que, en dicho ámbito, el parámetro más crítico y difícil de ajustar es la cantidad de feromonas que deben depositarse en base a una función diferencial. Adviértase que, dado que el mismo problema podría variar de forma dinámica, la cantidad de feromonas que una hormiga debe desplegar en función del escenario se antoja prácticamente impredecible, aunque si estimable. Una posible metodología para sobrepasar dicho obstáculo consiste en dotar a los formícidos de una suerte de sistema de aprendizaje *online*, en el que los artrópodos pueden consultar el estado global del escenario. Tal ventaja permitiría a las hormigas involucradas cuantificar el grado de calidad de la solución encontrada con respecto a sus homólogos, desplegando una cantidad de feromonas directamente proporcional a la misma.

Otro de los parámetros críticos en cuanto a la persecución del rendimiento óptimo por parte del ACO se corresponde con el número de hormigas a utilizar en el mismo. El número exacto de formícidos constituye un input que, en la mayoría de los casos, debe ajustarse de forma experimental. A pesar de que el ACO no presenta una variación extremadamente significativa en su rendimiento con respecto al número de formícidos usado, si presenta cierto impacto que debe ajustarse en caso de que la optimización del tiempo de respuesta sea una prioridad. Adviértase que aunque en la actualidad no existe ninguna fórmula que otorgue el número exacto de formícidos a involucrar en el algoritmo, los resultados prácticos instan a utilizar el mayor número de artrópodos posible. Este comportamiento presenta un problema, y es que, eventualmente, desembocará en problemas de rendimiento asociados, entre otros factores, a la *Ley de los Rendimientos Decrecientes*.

Encontrar el número exacto de hormigas requeridas para solucionar un problema

es, por tanto, un problema de fina granularidad. Desde el punto de vista matemático, es recomendable trazar una comparativa en el eje de coordenadas, que comparará el tiempo requerido por el ACO para converger en una solución frente al número de hormigas. Llegado cierto momento, se alcanzará un punto de estabilización, donde añadir una hormiga extra al problema no afectará el tiempo necesitado para hallar una solución de forma tan drástica como antes. Es reseñable que el número concreto dependerá, con casi total seguridad, del tipo de problema abordado por el ACO.

Tal y como ocurre con la mayoría de los algoritmos involucrados en la presente Tesis Doctoral, el Algoritmo de la Colonia de Hormigas presenta, a su vez, un amplio y jugoso elenco de variaciones, entre las que se destacan las siguientes:

- *Algoritmo de Colonia de Hormigas basado en Ranking:* En esta variante, las soluciones disponibles se clasifican en un *ranking* acorde a su longitud. Por otro lado, la cantidad de feromonas desplegadas en cada ruta es ponderada para la solución final, de tal forma que aquellas rutas más cortas poseen más feromonas que sus homólogas más largas.
- *Optimización basada en Colonia de Hormigas con Lógica Difusa:* Destacando que la Lógica Difusa (*fuzzy logic*) hace alusión a la variación del modelo numérico-algebraico en base dos, en la que un valor no solo puede variar entre cero y uno, sino a su vez en todos los valores intermedios, esta variante algorítmica pretende acelerar las habilidades de búsqueda pasando de un espacio discreto a su homólogo continuo.
- *Algoritmo de la Colonia de Hormigas Ortogonal Continua:* En esta variación, el mecanismo de depósito de feromonas pretende otorgar a los formícidos la posibilidad de hallar soluciones de forma conjunta, aumentando, por ende, la efectividad del algoritmo. Mediante el enfoque ortogonal introducido, los artrópodos disfrutan de una mayor capacidad de búsqueda al gozar de una visión panorámica del escenario, intercambiando de forma más directa información con sus homólogos, y no sólo tácitamente mediante el intercambio de feromonas.
- *Sistema Elitista de Formícidos:* De forma similar a los conceptos detallados en tal sentido en el epígrafe dedicado a la Computación Evolutiva, la variante en cuestión propone que la mejor solución global deposite feromonas especiales, en cada iteración. Tales feromonas serán aquellas que evidencien dónde se encuentra la mejor solución (que puede ser parcial en las primeras etapas del algoritmo) hallada hasta el momento.
- *Algoritmo de la Colonia de Hormigas basado en la técnica Min-Max:* Partiendo de la especificación de la cantidad mínima (*Min*) y máxima (*Max*) de feromonas, la variante en cuestión, que imita al Algoritmo *Min-Max*

clásico de la Inteligencia Artificial, pretende que únicamente la mejor iteración despliegue feromonas en un determinado camino. De esta forma, en la fase de inicialización del algoritmo todas las aristas del grafo son inicializadas con el valor Max, inicializándose de nuevo en caso de aparecer una situación *deadlock* o de interbloqueo.

Una vez definidas las principales características del Algoritmo de Colonia de Hormigas, así como sus variantes; ¿cuáles son los principales puntos de aplicación de este ámbito algorítmico? Una vez más, la versatilidad del algoritmo, unida a su sencilla facilidad de extrapolación, hace que estos estén casi únicamente delimitados por la imaginación. Se destaca que numerosos con base en el algoritmo en cuestión han sido adaptados a problemas dinámicos con variables reales, programación paralela y problemas de tipo estocástico. Además, se ha utilizado el Algoritmo de la Colonia de Hormigas para ejecutar la búsqueda de soluciones cuasi óptimas en el denominado Problema del Viajante (*Travelling Salesman Problem*), siendo este el primer problema abordado por el algoritmo que da nombre al presente epígrafe. Adviértase que dicho problema no es en absoluto baladí, pues es uno de los denominados *Problemas NP-Complejos*, de difícil solución computacional. (Este problema, que ha resistido tenazmente los esfuerzos de la comunidad matemática para solucionarlo de manera óptima, plantean fundamentalmente, la cuestión: Dada una lista de ciudades, así como las distancias entre cada par de ciudades, ¿cuál es la ruta posible más corta que visita cada ciudad exactamente una vez y retorna a la ciudad de origen?). La razón que justifica el sobresaliente comportamiento del Algoritmo de la Colonia de Hormigas en el ámbito del Problema del Viajante, radica en que aquella técnica algorítmica posee una ventaja competitiva sobre las aproximaciones restantes: en caso de que el escenario varíe de forma dinámica, el algoritmo puede continuar su ejecución de la misma forma, adaptando los cambios en tiempo real sin acusar ningún impacto negativo en su rendimiento. Como se detallará en posteriores epígrafes de la presente Tesis Doctoral, esta benigna característica impregna al Algoritmo de la Colonia de Hormigas de un especial e importante interés en el ámbito de los STPU.

En la misma línea, una de las principales aplicaciones actuales del Algoritmo de la Colonia de Hormigas se corresponde con el *Aprendizaje Máquina (Machine Learning)*, como demuestra [de Campos, 2002], más en concreto en su aplicación para Redes Bayesianas, así como para problemas adscritos a Big Data. Típicamente ambos ámbitos están íntimamente unidos, puesto que el *Aprendizaje Máquina* trabaja con ingentes cantidades de información. Principalmente, el algoritmo en cuestión se aplica a tareas con un factor común claramente delimitado: la agrupación de objetos con similares características. Por otro lado, se destaca que la optimización de estructuras de hormigón armado es, a su vez, un problema en el que el Algoritmo de la Colonia de Hormigas goza de un protagonismo excepcional: Dado que dicha técnica algorítmica se comporta de

manera efectiva independientemente de la aleatoriedad del entorno, el azar presente en las estructuras atómicas del hormigón armado no supone un impacto excesivamente pernicioso en el rendimiento computacional, siendo el algoritmo en cuestión uno de los mayores aliados contemporáneos en dicha línea de investigación.

De forma paralela, es destacable que a la vez que existen variantes algorítmicas extrapoladas de forma directa del Algoritmo de la Colonia de Hormigas, existen metodologías relacionadas en mayor o menor medida, a saber:

- *Algoritmos Genéticos (GA)*: Exhaustivamente descritos en el epígrafe anterior, los GA presentan evidentes similitudes con el algoritmo referido a los formícidos, aunque una clara diferencia: Los GA sustentan un grupo de soluciones en vez de sólo una. El proceso de encontrar mejores soluciones (idealmente óptimas) constituye un símil con el proceso evolutivo de los seres vivos. Tal y como se especifica en el epígrafe precedente, en un GA se descartan las soluciones que no se consideran relevantes, mientras que las restantes se combinan en base a los operadores genéticos a fin de hallar un subconjunto mejorado de cromosomas en el genotipo.
- *Búsqueda Tabú (BT, del inglés TS; Taboo Search)*: Tipo de búsqueda a caballo entre la Algorítmica Genética y los Algoritmos de la Colonia de Hormigas, en la que se pretende atravesar el espacio de soluciones mutando progresivamente una solución parcial. TS toma su nombre de una de sus principales características: a fin de evitar la existencia de ciclos en la búsqueda, mantiene una lista negra o tabú con soluciones parciales o completas. De esta forma, el algoritmo funciona en base a una simple premisa: Se prohíbe computar una solución en la que exista algún elemento presente en la lista tabú, la cual se actualiza progresivamente en las distintas iteraciones del algoritmo.
- *Sistema Inmune Artificial (AIS; Artificial Immune System)*: De la misma forma que el Algoritmo de la Colonia de Hormigas imita el comportamiento de dichos formícidos en la búsqueda de alimento, la presente técnica algorítmica toma su razón de ser en la extrapolación de los procesos acontecidos en los Sistemas Inmunológicos de los seres vertebrados. De esta forma, AIS constituye una algorítmica de búsqueda que, al igual que ocurre con el ACO, halla su principal fortaleza en su elevado nivel de adaptabilidad.
- *Optimización por Enjambre de Partículas (PSO; Particle Swarm Optimization)*: Los algoritmos adscritos al ámbito de la PSO, de gran similitud con ACO, serán, precisamente, objetivo de la investigación

detallada en el epígrafe siguiente, por lo que su detalle exhaustivo será reservado para el mismo dada su entidad.

Cabe destacar, a modo de conclusión respecto a las variaciones del ACO, que el algoritmo en su versión más puramente clásica presenta un mejor rendimiento en comparación a las entidades anteriormente descritas, por lo que la inclusión de las mismas únicamente se recomienda si la índole del problema se circunscribe de forma cuasi total a estas. Con el fin de demostrar empíricamente dicha aseveración se presenta la siguiente tabla, que compara el rendimiento del ACO clásico con las variaciones más populares; la Búsqueda Tabú o BT y el Recocido Simulado o RS (del inglés *Simulated Annealing*; SA). Advuértase que, para el mismo escenario, el Algoritmo de la Colonia de Hormigas en su versión clásica presenta un mejor valor global, así como a su vez una media similar a BT y un valor de Desviación Estándar significativamente menor con respecto a sus competidores más directos (los mejores valores aparecen resaltados):

	Mejor tiempo de búsqueda	Media (s) 1000 ejecuc.	Desviación Estándar
ACH	420	420.4	1.3
Búsqueda Taboo	420	420.6	1.5
Recocido Simulado	422	459.8	25.1

Figura XXXIV: Comparativa de rendimiento entre el Algoritmo de la Colonia de Hormigas y sus principales variantes

Es digno de mención que el experimento está ejecutado en un *MacBook Pro* con procesador *Intel Core i7* de cuatro núcleos, a 2,9 GHz con *Turbo Boost* de hasta 3,9 GHz, 8 MB de caché de nivel 3 compartida y un *SSD PCIe* integrado de 512 GB.

Una vez enumeradas y descritas las variaciones del ACO, así como las ramas algorítmicas que toman dicho algoritmo como inspiración principal, es importante advertir al lector de una, en parte peligrosa, tendencia actual referida al concepto del ACO: Muchos algoritmos actuales aluden al término ACO como su raíz conceptual, a pesar de que distan mucho de presentar características análogas a las de la algorítmica formícida descrita en el presente epígrafe. En la práctica, ciertos autores consideran oportuno llamar Algoritmo de Colonia de Hormigas siempre y cuando el concepto de estigmergia esté involucrado en sus algoritmos. (Es decir, los agentes presentes en el software intercambian información en base a su entorno.) Tal y como se extrae de numerosos escritos de Marco Dorigo, creador de dicha rama algorítmica, esta asociación es errónea, puesto que

prácticamente impredecible el modelo teórico que adoptará cada ACO, por lo que se recomienda llevar a cabo una investigación más experimental que puramente teórica.

- Existe una curiosa característica extraída de los ACO, que constituye una suerte de híbrido entre ventaja y desventaja; La *convergencia* del algoritmo es un suceso seguro, aunque no obstante el tiempo de convergencia es incierto. Por las razones argumentadas en el punto anterior, este es además de prácticamente imposible de calcular *a priori* con exactitud desde un enfoque teórico.
- Los ACO tienden a presentar bifurcaciones en función de la naturaleza del problema a resolver, factor que dota de complejidad a la delimitación del algoritmo. Por ejemplo, los problemas de enrutamiento dinámico de redes necesitan soluciones para condiciones cambiantes, por lo que se debe hacer especial énfasis en una evaluación efectiva de los caminos alternativos, mientras que en la clase de complejidad NP-Completo se necesitan soluciones de óptima calidad con notable celeridad, siendo el área donde se debe llevar a cabo una especial incisión.
- La implementación adscrita a los ACO no es, en absoluto, trivial. A pesar de que el comportamiento de los artrópodos involucrados en el algoritmo está claramente delimitado, el elevado número de factores a implementar impacta negativamente en la dificultad asociada a la programación del ACO, que puede ser extremadamente elevada en ciertos casos. Los numerosos factores a implementar varían en un amplio rango, desde la actualización de los rastros de feromonas hasta los cambios globales y locales que acontecen en el algoritmo. Por otro lado, y como se relata previamente en el presente epígrafe, existe un gran número de variaciones del ACO para explotar diferentes características, por lo que la elección de dicha variación se antoja decisiva, ya que impactará directamente en las etapas de diseño e implementación del software. El número de formícidos a utilizar en la colonia virtual que posee el algoritmo constituye, como se especifica previamente en el presente epígrafe, un crítico valor que impactará directamente en el rendimiento, y cuya delimitación debe llevarse a cabo mediante una aproximación experimental al no existir ninguna fórmula o herramienta matemática para delimitarlo en la actualidad.

2.1.4 Particle Swarm Optimization

Desde los albores de la ciencia, numerosos científicos se han visto intrigados por un movimiento tan elegante como óptimo presente en la naturaleza: La armoniosa sincronía de las bandadas de aves o los bancos de peces, capaces de desplazarse sin ni siquiera rozarse a pesar de los cientos de seres que componen dichos conjuntos. Fruto de la investigación científica, no sólo se descubre que tales criaturas son capaces de desplazarse en conjunto sin ni siquiera rozarse, sino que a su vez presentan ciertos patrones de *enjambre* en su búsqueda de alimento.

Más en concreto, destaca el hiperbólico interés de Grenander y Heppner en su opus magnum *A stochastic nonlinear model for coordinated bird flocks* [Heppner et al., 1990], en el que ambos zoólogos sintetizan su investigación referida a las directrices, ocultas por la naturaleza, que marcan el movimiento asíncrono de las bandadas de aves, cambiando en ocasiones su dirección de forma repentina, escapando de depredadores y reagrupándose tácitamente, entre otras fascinantes habilidades. Destaca, de forma paralela, el brillante trabajo de Reynolds en la misma línea, *Flocks, herds and schools: a distributed behavioral model* [Reynolds, 1987], focalizado a su vez en la intrigante coreografía de las aves en vuelo.

Los científicos anteriormente citados, cuyas obras serían vitales en la posterior definición de la metaheurística *Particle Swarm Optimization* (Optimización por Enjambre de Partículas), defendían firmemente la idea de que los procesos locales que acontecen en dichos eventos, así como los modelados por los autómatas celulares, podrían ser la base de la dinámica de grupo, de tan impredecible índole, del comportamiento social de aves. Tales modelos dependen intrínsecamente de la manipulación de las distancias entre individuos, de forma se empieza a sospechar que la sincronía de las bandadas de aves puede ser el resultado del esfuerzo llevado a cabo por las mismas para mantener una distancia óptima entre sus homólogas. A tenor de sus investigaciones, la terna de científicos anteriormente citada comenzó a elucubrar una idea que el paso del tiempo reforzó como en absoluto descabellada; Las bandadas de aves no son los únicos seres que presentan este comportamiento, presente a su vez en las manadas de seres terrestres, los bancos de animales acuáticos e incluso en los humanos. Baste citar, a modo de ejemplo, la estrategia de reagrupación, cuasi militar, que adoptan los *Thunnus Albacares* (Atún común) al enfrentar una hipotética amenaza:

Figura XXXV: Banco de *Thunnus Albacres* reagrupándose frente a una hipotética amenaza

Precisamente en referencia a los bancos de estos vertebrados acuáticos, el socio biólogo Edward Osborne Wilson establece, en su obra *Sociobiology: The new synthesis* [Wilson, 1975] que: *"In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantages of competition for food items, whenever the resource is unpredictably distributed in patches"*. Lo que se traduce como: *"Al menos en la teoría, los miembros individuales de un rebaño pueden beneficiarse de los descubrimientos y previa experiencia de sus homólogos durante la búsqueda de alimento. Esta ventaja se torna decisiva, prevaleciendo sobre las desventajas de la competición por los alimentos, cuando el recurso está distribuido de forma imprevisible en parcelas."*. Adviértase que dicha aseveración sugiere un factor revolucionario en su ámbito cronológico, pues apunta a que la compartición de información entre individuos de la misma especie provee una vital ventaja evolutiva. Como se demostrará a lo largo del presente epígrafe, la hipótesis formulada por Wilson en 1975 supone el pilar fundamental del Paradigma *Particle Swarm Optimization*.

Cabe destacar con respecto a dicha hipótesis, que, guardando las distancias, el modelo de comportamiento humano presenta ciertas características también adscritas a las demostradas por las bandadas de aves y los bancos de peces. Una de las más importantes diferencias con respecto a las mismas es el concepto de *abstracción*: Mientras las aves y los vertebrados acuáticos ajustan estratégicamente su desplazamiento para evitar la peligrosidad de los depredadores, buscar alimento o pareja y optimizar su temperatura corporal (se recuerda que los peces son seres ectotérmicos, pues regulan su temperatura a

tenor del medio ambiente), el ser humano no sólo ajusta su movimiento físico, sino también su comportamiento *cognitivo* o experimental. El ser humano no tiende a caminar en grupo y cambiar el sentido de la caminata de forma estrictamente sincronizada, pero en contraposición sí tiende a comportarse de la misma forma que sus homólogos. Adviértase que existe una característica subyacente pero vital en cuanto a los aspectos referidos a la simulación informática de este comportamiento, y es que dos individuos humanos pueden chocarse en un escenario mientras que este hecho podría ser fatal en el ámbito de una bandada de aves, por lo que dos pájaros no pueden, obviamente, compartir el mismo punto en el espacio sin evitar una colisión.

Englobando los conceptos anteriormente citados, es lícito inferir que los seres humanos transitan en un espacio *multidimensional y libre de colisiones*, mientras que el resto de conjuntos de criaturas, tales como las bandadas de aves y los bancos de peces, no pueden gozar de la misma libertad. Esta sutil, pero crucial diferencia impregnará, tal y como se verá a continuación, la naturaleza del Paradigma *Particle Swarm Optimization*. Dicho algoritmo, inicialmente atribuido a Kennedy, Eberhart y Shi en sus obras *Proceedings of IEEE International Conference on Neural Networks* y *Proceedings of IEEE International Conference on Evolutionary Computation*, fue concebido originalmente con el fin de simular el comportamiento gregario de las especies, entendiendo como tal una representación formal del movimiento de los seres que componen una bandada de pájaros o un banco de peces.

Llegados a este punto, se está en posición para abordar la definición formal del algoritmo que bautiza al presente epígrafe, que, junto al Algoritmo de la Colonia de Hormigas, cierra el conjunto de técnicas adscritas a la rama de la Inteligencia Artificial denominada *Swarm Computing*: Se denomina *Particle Swarm Optimization* (Optimización por Enjambre de Partículas; PSO en lo sucesivo), al Paradigma computacional que pretende optimizar un problema de forma meta-heurística, es decir, mediante el intento iterativo de mejorar una solución candidata con respecto a una medida de calidad estipulada *a priori*. Adviértase que, de forma similar a los Algoritmos Genéticos, PSO optimiza un problema a partir de una población de soluciones candidatas, típicamente partículas en el espacio, desplazándolas a través del espacio de búsqueda en base a las premisas de una fórmula matemática que involucra la posición y velocidad de las partículas. Como puede intuirse, la técnica mimetiza el comportamiento conjunto de las agrupaciones de seres vivos citadas a lo largo del presente epígrafe, en el que el movimiento de cada individuo o partícula se ve influenciado por la posición local mejor conocida, a la vez que, de forma paralela, la mejor posición conocida a nivel global por la colonia de criaturas. Es reseñable que tal posición global es pertinentemente actualizada por la mejor posición conocida por los individuos del conjunto, factor que guiará al enjambre a desplazarse en pos de la mejor solución global.

Es digno de mención que el Paradigma PSO adopta una ínfima cantidad de supuestos en su proceso de ejecución, explorando un amplio espacio de soluciones. A pesar de ello, PSO es una meta-heurística, por lo que no se puede asegurar de forma tajante que el algoritmo encuentre la solución óptima del problema en todos los casos. De forma más específica, PSO no utiliza el gradiente del problema abordado, lo que se traduce en que esta técnica no necesita que el problema en cuestión sea diferenciable, tal y como ocurre con las metodologías de optimización típicas tales como El Descenso por Gradiente o los métodos Cuasi-Newtonianos. Por consiguiente, PSO puede utilizarse, con un elevado índice de acierto, en los problemas de optimización especialmente irregulares, en los que existe cierto ruido ambiente o en los que constan de carácter dinámico, cambiando a lo largo del tiempo. A pesar de que los computadores son sistemas altamente eficientes, realizando cálculos a una velocidad inalcanzable incluso para las más avezadas mentes matemáticas humanas, presentan una perniciosa característica: la inflexibilidad y problemática variada cuando se enfrentan a problemas del mundo real, especialmente aquellos difusos, que presentan cierto ruido ambiente o que están impregnados de un carácter multimodal motivado por su inflexible construcción. Los sistemas de índole natural han evolucionado a lo largo de la historia en pos de la resolución de tales problemas, presentando, en su gran mayoría, una característica común: La combinación de elementos simples produce, al ejecutar una acción en términos conjuntos, un comportamiento complejo. Adviértase que dicha característica ha inspirado un amplio espectro de paradigmas computacionales de corte natural, que han sido aplicados satisfactoriamente en casuísticas donde las técnicas computacionales clásicas presentan un rendimiento insatisfactorio. En esta línea, y tal y como se detallará a lo largo del presente epígrafe, el paradigma PSO ha permitido su aplicación a un cuantioso número de problemas caracterizados por presentar un dominio de optimización complejo [Kennedy et al., 1995], ofreciendo una alternativa a los paradigmas de computación de corte evolutivo que pueden aplicarse en tales dominios.

Es digno de mención que muchos de los paradigmas de computación naturales presentes en la actualidad comparten ciertas características desde el punto de vista conceptual y de entrada, pues, por ejemplo, deben lidiar con información que frecuentemente es incompleta o está impregnada de ruido, tener la habilidad de resolver problemas adscritos al campo de la Combinatoria matemática, especialmente intensiva en cómputo, adecuarse a características emergentes donde el comportamiento complejo del sistema en su conjunto contrasta con la simplicidad de los agentes que lo componen, etc. Con respecto a tales características, se destaca que las arquitecturas de procesamiento simples, inspiradas por procesos naturales, proveen formas de computación ciertamente útiles. De la misma forma que el paradigma PSO, estas surgen, típicamente, del comportamiento descentralizado clásico, característicos de los sistemas auto-

organizados. A pesar de las numerosas bondades de este tipo de sistemas, su principal inconveniente se asocia con su indeterminismo; tales paradigmas ofrecen soluciones aproximadas a problemas para los cuales una estrategia analítica o de corte numérico tradicional no es factible, pero siempre a costa del reto, en absoluto trivial, que respecta a su diseño y validación.

De esta manera, se está en disposición de presentar formalmente el funcionamiento del algoritmo que da nombre al presente epígrafe. De la misma forma que ocurre en el ámbito de los algoritmos genéticos, tal funcionamiento parte de un input formado por una población de partículas (soluciones candidatas), denominada típicamente nube o enjambre. Las anteriormente mencionadas partículas llevarán a cabo un desplazamiento a lo largo del espacio de búsqueda conforme a los criterios matemáticos cuyos detalles serán pormenorizadamente desmenuzados seguidamente.

Con respecto a las propias partículas o soluciones candidatas, destaca que el movimiento de cada partícula depende directamente de su mejor posición obtenida. Así, el movimiento de las partículas será directamente dependiente de la emergencia de mejores posiciones, que orientarán los movimientos de las soluciones candidatas. Como es frecuente en la Algorítmica Natural, este comportamiento será iterado hasta que se halle una solución aceptable, que satisfaga ciertos criterios establecidos a priori. Una vez más ha de hacerse especial hincapié en que la probabilidad de no encontrar una solución no es, en absoluto, despreciable.

De esta forma, desde un punto de vista más puramente matemático, se establece que, siendo f una función matemática de coste que se pretende minimizar [Banks et al., 1981], tal que:

$$f : R^n \rightarrow R$$

Quedando f definida, ésta toma como input una solución candidata, que será representada por un array de números reales, mientras que el output será un número perteneciente al conjunto $\mathbb{R}$, indicativo del valor de la función objetivo para la solución candidata obtenida. Así, las mejores posiciones serán aquellas que se corresponden con los mejores valores de la función objetivo f . En este caso, dado que la naturaleza del problema pretende la minimización de f , se busca el hallazgo de una solución a que, siendo b cualquier elemento en el espacio de búsqueda, verifique:

$$f(a) \leq f(b)$$

De esta forma, al respetarse tal criterio, se estaría en disposición de afirmar que a es el mínimo global. Adviértase que el proceso inverso puede aplicarse en problemas de maximización de funciones, que a su vez son susceptibles de ser abordados mediante el algoritmo PSO.

Una vez introducidos los conceptos fundamentales referidos al algoritmo, se procederá a la descripción de los pormenores referidos a funcionamiento: Sea S el número de partículas en la nube, teniendo cada una de estas soluciones una posición $x_i \in R^n$ en el espacio de búsqueda y una velocidad $v_i \in R^n$. Sea p_i la mejor posición conocida de la partícula i , g la mejor posición global conocida por la población de partículas en su totalidad, y ω , φ_p y φ_g los parámetros del algoritmo, (sobre los que se ahondará seguidamente) entonces un algoritmo PSO define su funcionamiento como:

- Para cada partícula perteneciente a la población, $i = 1, \dots, S$
 - o Inicializar la posición de la partícula a partir de una variable aleatoria X_i verificando que $X_i \sim U(b_{lo}, b_{up})$. De esta forma, X_i sigue una Distribución estadística de tipo Uniforme, siendo b_{lo} el límite inferior y b_{up} su homólogo superior.
 - o Inicializar la mejor posición conocida de la partícula a su posición inicial: $p_i \leftarrow x_i$
 - o Si se verifica que $f(p_i) < f(g)$, entonces actualizar la mejor posición global conocida con el nuevo punto óptimo hallado: $g \leftarrow p_i$
 - o Inicializar la velocidad de la partícula $v_i \sim U(b_{lo} - b_{up}, b_{lo} - b_{up})$
- Mientras el criterio de parada no se cumpla, repetir:
 - o Para cada partícula, $i = 1, \dots, S$
 - Para cada dimensión $d = 1, \dots, n$
 - Elegir una dupla de números aleatorios tal que: $r_p, r_g \sim U(0, 1)$
 - Actualizar la velocidad de la partícula en función de la fórmula:

$$v_i \leftarrow \omega v_{i,d} + \varphi_p r_p (p_{i,d} - x_{i,d}) + \varphi_g r_g (g_d - x_{i,d})$$
 - Actualizar posición de la partícula: $x_i \leftarrow x_i + v_i$
 - Si se verifica que $f(x_i) < f(p_i)$, entonces:
 - Actualizar la mejor posición conocida de la partícula: $p_i \leftarrow x_i$
 - Si $f(p_i) < f(g)$, actualizar la mejor posición global: $g \leftarrow p_i$
- Retornar g como la mejor solución encontrada.

Es digno de mención que la selección de los parámetros ω , φ_p y φ_g adscritos al ámbito de la PSO se antoja vital, pues impactará directamente en el rendimiento del algoritmo. Si se toma por referencia que la función a optimizar mediante PSO se asocia con una hipersuperficie de dimensionalidad equivalente al número de variables de búsqueda (o parámetros de optimización), la irregularidad de dicha hipersuperficie dependerá del problema en concreto. Como conclusión directa a la anterior aseveración, la exhaustividad de la búsqueda será directamente proporcional a su calidad, que estará íntimamente relacionada con los parámetros ajustados. Como ejemplo, se presenta a continuación una superficie a modo de

gráfica de dos dimensiones que ilustra el rendimiento de una variante PSO ante un conjunto de pruebas aplicado a problemas de parámetro dual [Shi et al., 1998]:

Figura XXXVI: Hipersuperficie ilustrativa del rendimiento de PSO ante un benchmark de problemas biparamétricos

Con el fin de obtener soluciones que presenten una superficie prácticamente plana, se necesita un número ínfimo de partículas e iteraciones. Por otro lado, en caso de que se pretenda justo el comportamiento opuesto, una hipersuperficie de índole irregular necesita de un mayor número de partículas e iteraciones, por lo que puede inferirse que la rugosidad de la hipersuperficie es directamente proporcional al número de partículas e iteraciones. Adviértase puede establecerse un símil entre este escenario virtual y un escenario real: el de la ganadería trashumante, en la que pequeños rebaños pueden alimentarse mediante terrenos que no son especialmente frondosos en cuanto a vegetación (siendo un óptimo local), mientras que un gran rebaño deberá atravesar, teóricamente, terrenos más abruptos para hallar prados más frondosos (óptimo global).

Tal y como se ha desarrollado previamente, la precisión en el ajuste de los parámetros involucrados en el algoritmo será un factor determinante de cara al rendimiento de la técnica. Precisamente en esta línea, es reseñable que el algoritmo PSO transita entre dos grandes comportamientos:

- Comportamiento *Explotador*: Búsqueda orientada al ámbito local con el fin de acercarse a un punto óptimo, que, en la mayoría de los casos, será

local. Presenta una gran celeridad, más difícilmente encontrará la mejor solución posible.

- Comportamiento *Explorador*: En contraposición al anterior, búsqueda orientada al ámbito global, pretendiendo hallar un óptimo global. Más longeva en tiempo de cómputo que su homóloga, pero más efectiva a su vez a la hora de encontrar la mejor solución existente.

De esta forma, es lícito inferir que los parámetros ω , φ_p y φ_g han de ajustarse en pos de salvaguardar un equilibrio razonable entre la *explotación* y la *exploración*, dependiendo a su vez del carácter de la problemática a resolver. Así, podrá evitarse la convergencia prematura hacia un óptimo local, a la vez que se asegura un aceptable ratio de convergencia hacia el óptimo global.

Por otro lado, se destaca que numerosos autores conciben el comportamiento del algoritmo PSO de una forma opuesta [Poli, 2007], entendiendo que el comportamiento de la técnica no se ajusta con especial bondad a la optimización del rendimiento, especialmente en el escenario adscrito a los espacios de búsqueda multidimensionales, donde los problemas de optimización típicamente son de naturaleza continua y variante en función del tiempo, a la vez que impregnados de una capa de ruido. A tenor del fundamento anteriormente expuesto, este conjunto de autores defiende la búsqueda de algoritmos extraídos de la técnica PSO y parámetros que logren alcanzar un más que aceptable rendimiento, independientemente de cómo el comportamiento en enjambre puede ser interpretado en términos de explotación y exploración. La naturaleza dispersa y borrosa de algunos aspectos referidos al algoritmo PSO, dificulta la tarea de decantarse por una mejor alternativa al respecto, aunque esta decisión puede realizarse experimentalmente en pos de la *convergencia*. Más ejemplos al respecto pueden comprobarse en [Pso, 2018] y [Swarm, 2018].

Se entiende por *convergencia* a la característica que se presenta en el escenario en el cual todas las partículas han rodeado a un punto concreto en el espacio de búsqueda, que puede o no ser el punto óptimo. En concreto, la convergencia denota el acercamiento a un óptimo local donde todos los mejores valores p (personales) se encuentran, o, de manera alternativa pero análoga, la mejor posición conocida por el enjambre, g .

Específicamente, se referencia al lector al razonamiento investigado en "*An Analysis of Particle Swarm Optimizers*" [Bergh, 2001], en el que los análisis pertinentes han desembocado en ciertas guías de selección de los parámetros ω , φ_p y φ_g que posibilitan la convergencia hacia un punto óptimo frecuentemente global, previniendo la divergencia de las partículas que componen el enjambre hacia óptimos locales. No obstante, es digno de mención que tales razonamientos han sido objeto de crítica por "*Simplifying particle swarm*

optimization" [Pedersen, M, 2010], pues las inferencias expuestas en aquel estudio parten de la base de que el enjambre se compone de, únicamente, una partícula. Esta simplificación extrema, unida a la carencia de variables estocásticas, motivan que la mejor posición conocida por una partícula, p , y la mejor posición conocida por el enjambre, g , prevalezcan inmutables a lo largo del proceso de optimización, obviando la exploración de un espacio de optimización hipotética en absoluto desdeñable. De forma paralela, "*Particle Swarm Convergence: Standardized Analysis and Topological Influence*" [Cleghorn, 2014] demuestra que la simplificación llevada a cabo por Van den Bergh no afecta en gran medida a los límites encontrados por dicho estudio paramétrico en aquellos escenarios donde el enjambre es convergente.

A pesar de los numerosos estudios que toman por objeto la convergencia hacia un óptimo local en el algoritmo PSO, este necesita aún una cierta modificación, desconocida hasta la fecha, que garantice el hallazgo de un óptimo local en todos los casos, esbozando la puerta a una vía de investigación en el ámbito algorítmico de PSO. Determinar las capacidades de convergencia de los distintos algoritmos PSO y sus parámetros depende, por tanto, de resultados de carácter empírico en la actualidad, siendo numerosas las teorías, finalmente infructuosas, al respecto. No obstante, y aunque posiblemente sea la excepción que confirma la regla anteriormente relatada, una prometedora línea impregna de un estilo revolucionario al algoritmo PSO, basándose en un principio de *aprendizaje ortogonal* [Kennedy et al., 2001]; Pretendiendo explotar la posesión de la información existente entre los valores p y g , (mejor posición conocida por la partícula y mejor posición conocida por el enjambre, respectivamente), se busca dar forma a un tipo de convergencia que sea efectiva para cualquier topología PSO dada. La técnica de aprendizaje ortogonal pretende incrementar el rendimiento del algoritmo PSO en términos globales, entendiendo como tales una mayor calidad de la solución (en caso de no ser global), una convergencia, idealmente global, más rápida y una robustez incrementada. Desgraciadamente, los estudios existentes en la actualidad no proveen una evidencia teórica suficiente como para demostrar, empíricamente, sus beneficiosos argumentos.

Al igual que el ajuste de los parámetros y la relación entre los valores p y g es de vital importancia en el ámbito del algoritmo PSO, el sesgo involucrado en el mismo lo es a su vez. En términos generales, se entiende por sesgo la oblicuidad o torcimiento de un elemento hacia un extremo. En el caso concreto del PSO, y dado que dicho algoritmo funciona en base a dimensiones, la solución se encuentra con mayor celeridad cuando reside en los ejes del espacio de búsqueda, en una diagonal, e incluso con una mayor velocidad si está en el centro del mismo. No obstante, tal y como la lógica apunta, este escenario no es frecuente.

Una aproximación común para lidiar con la problemática adscrita a la presencia

del sesgo, es la de modificar el algoritmo PSO de forma que deje de ser sensible al sistema de coordenadas. En esta línea, destaca el razonamiento presente en "*An analysis of the velocity updating rule of the particle swarm optimization algorithm*" [Bonyadi et al., 2014]. A pesar de que es posible minimizar el impacto negativo que supone la presencia del sesgo, esta ventaja se obtiene a costa de un incremento de la complejidad computacional en el algoritmo, que asciende a $O(n^2)$ donde n se corresponde con el número de dimensiones. Este incremento se traduce en una difícil transición algorítmica hacia una optimización a gran escala.

Finalmente, en aquellos aspectos referidos al sesgo se refieren, es digno de mención que la única alternativa eficaz existente en tal sentido en el ámbito de los PSO es la propuesta en "*A locally convergent rotationally invariant particle swarm optimization algorithm*" [Bonyadi, M. 2014], que sorpresivamente no es sensible a la rotación de las coordenadas mientras mantiene su convergencia local. El método citado en dicha obra ofrece un rendimiento algorítmico más que aceptable, mientras su invariancia rotativa y su convergencia local han sido probadas desde el punto de vista matemático.

Habiendo descrito los pormenores del algoritmo PSO en cuanto a sus aspectos paramétricos e internos se refiere, destaca la importancia de la topología del enjambre vista como un conjunto, así como los aspectos referidos a la interacción entre partículas. El algoritmo PSO clásico queda fácilmente *atrapado* en la convergencia hacia un mínimo local, pero dicha convergencia prematura puede ser evitada mediante el cese del uso de g (mejor posición conocida del enjambre) en detrimento de la mejor posición conocida, l , del subenjambre posicionado alrededor de la partícula en movimiento. Tal subenjambre puede elegirse en base a un criterio geométrico, como puede ser el delimitado por las n partículas más cercanas, o, más frecuentemente, por un criterio social. Tal criterio puede adoptarse como el que propone la selección de las partículas independientes de cualquier tipo de distancia, caso en el que dicha variante del algoritmo PSO alcanzará valores locales aceptables.

Partiendo de la suposición de que existe un enlace entre cada partícula y su vecindario, es lícito realizar una extrapolación de la topología que resulta en la conciencia del enjambre como si de un grafo se tratara. Dicha aplicación desemboca en la posibilidad de utilizar la potencia de la *Teoría de Grafos*, en desarrollo desde el siglo XVIII. La aplicación de esta estructura matemática permite, a su vez, comprender el enjambre en términos de *topologías de red*. Típicamente, el enjambre se considera una topología en anillo, en la que se verifica que cada partícula consta de la vecindad de dos homólogas. Es destacable que la elección de la topología puede adecuarse a la problemática concreta que se pretende resolver, así como a la índole del escenario en

cuestión. A continuación figuran varias topologías de red, inspiradoras de las numerosas posibilidades de interpretación del enjambre:

Figura XXXVII: Topologías de red susceptibles de aplicación al enjambre adscrito al algoritmo PSO

Como se detalla en varias secciones del presente epígrafe, la técnica PSO estándar utiliza el *flag gBest* puesto que las partículas involucradas en el algoritmo residen, normalmente, en una topología en estrella. Este factor significa que todas las partículas están conectadas entre sí, tal y como puede apreciarse en el área izquierda de la siguiente figura:

Figura XXXVIII: Topología en estrella (izq.) y en anillo (der.)

No obstante, existe otra importante topología, basada en el modelo de enjambre *lBest* y típicamente referida como *topología local en forma de anillo*, que puede observarse en la sección derecha de la figura anterior. Esta constituye, posiblemente, la mayor variación significativa del algoritmo PSO original, y es de hecho propuesta por una de las más tempranas publicaciones de Kennedy y Eberhart (1995). Independientemente de su longevidad, es reseñable que esta última topología no ha disfrutado de tanta popularidad entre la comunidad, y aunque investigaciones originales como [Engelbrecht, 2013] han demostrado que dicha topología presenta un rendimiento menor en PSO clásico, ha quedado en evidencia recientemente que los enjambres *lBest* presentan mejores resultados cuando son aplicados a conjuntos de problemas estándar, en conjunción con otras mejoras del algoritmo [Kennedy, 2006]. Así, es reseñable destacar, a título ilustrativo, que la topología local únicamente afecta a la siguiente ecuación, propia y fundamental de PSO y pormenorizadamente detallada previamente:

$$v_i(t + 1) = v_i(t) + c_1 * r_1 * (p_{Best} - x_i) + c_2 * r_2 * (g_{Best} - x_i)$$

Una vez descritos exhaustivamente el comportamiento y los pormenores referidos al algoritmo PSO, es recomendable atender a los distintos tipos de variación y simplificación existentes sobre el mismo. En primer lugar, destaca la prolífica línea existente dedicada a la simplificación de PSO, que defiende la necesidad de simplificación del algoritmo existente. Más en concreto, dicha línea aboga por la reducción del algoritmo PSO hasta el mayor límite posible, velando, en todo momento, por la minimización del impacto en el rendimiento, que idealmente se mantendrá inmutable. Adviértase que, de conseguir dicha meta satisfaciendo la máxima anterior, se lograría asestar un golpe, posiblemente letal, al problema del ajuste de los parámetros ω , φ_p y φ_g , descrito en secciones anteriores del presente epígrafe. Por otro lado, el ajuste de dichos parámetros podría ser ajustado de manera más coherente y homogénea en distintos problemas de optimización, incrementando notablemente la escalabilidad y versatilidad del algoritmo PSO.

En la misma línea que el razonamiento anterior, persiguiendo la simplificación del algoritmo PSO, destaca la posibilidad de eliminar la presencia de técnicas metaheurísticas en el mismo. Esta razón se debe a que tales técnicas únicamente presentan una eficacia demostrada empíricamente mediante el desarrollo de experimentos computacionales en un número finito de problemas de optimización. Por tanto, una metodología de corte metaheurístico como la presente en el algoritmo PSO carece de la posibilidad de demostrar su correctitud, factor que incrementa el riesgo de la caída en errores en las etapas

de diseño e implementación del software. A su vez, la inicialización de las velocidades presentes en las partículas podría requerir de valores de entrada adicionales, siendo un claro ejemplo de dicha técnica el método *Accelerated Particle Swarm Optimization* (APSO; Optimización de Enjambre de Partículas Acelerado), que destaca por la inexistencia del concepto de velocidad en las partículas, pudiendo acelerar notoriamente la velocidad de convergencia del algoritmo en un amplio elenco de escenarios.

Una vez introducido el concepto de simplificación algorítmica referido a PSO, se presentan las dos grandes variantes, en términos de simplificación, alusivas a dicha técnica:

- PSO Combinatoria, Discreta y Binaria (*Binary, Discrete and Combinatorial PSO*): Puesto que el fundamento matemático que sustenta a la PSO, definido previamente en el presente epígrafe, está preparado para su uso con números reales, un método común utilizado en la resolución de problemas adscritos al entorno discreto consiste en el proceso de *mapping* del espacio de búsqueda discreto en un dominio continuo. De esta forma, se posibilita aplicar un algoritmo PSO típico, para posteriormente, en la etapa de *finalización* adicional del algoritmo, traducir el resultado. Dicha traducción puede ser, en su variante más simplificada, un redondeo de valores que satisfaga los criterios pertinentes.
- PSO de Optimización Multi-Objetivo (*Multi-objective optimization PSO*): Dado que el algoritmo PSO ha sido aplicado satisfactoriamente a numerosos problemas multi-objetivo, en el que la función de comparación toma la Dominancia de Pareto en cuenta llegado el momento de desplazar las partículas que componen el enjambre, la optimización orientada a la mejora de más de una variable componen una prometedora línea de investigación en el ámbito de los PSO. Se recuerda al lector que la *Dominancia de Pareto* establece que, dado un vector $u = (u_1, \dots, u_k)$, se dice que domina a otro vector $v = (v_1, \dots, v_k)$ si y sólo si:

$$\forall i \in \{1, \dots, k\}, \quad u_i \leq v_i$$

$$\exists i_o \in \{1, \dots, k\} \mid u_{i_o} < v_{i_o}$$

Definidas las distintas simplificaciones posibles en el algoritmo PSO, es pertinente detenerse en las variantes disponibles. A modo de ejemplo, cabe destacar las distintas vías de inicialización de partículas y sus pertinentes velocidades que proponen diversos autores, la variante que únicamente actualiza los valores p y g una vez el enjambre entero ha sido actualizado, etc. Además de la dupla anteriormente citada, existen dos grandes variantes dignas de mención:

- PSO de *Alivio Prematuro (Alleviate Premature PSO)*: La variante en cuestión del algoritmo PSO, toma su nombre de su principal objetivo: minimizar (aliviar) la convergencia prematura en la medida de lo posible. Tal objetivo puede conseguirse mediante la variación del movimiento de las partículas que componen el enjambre, así como mediante el uso de múltiples enjambres de forma paralela. Se destaca que esta técnica, de mayor enjundia, ha dado el pistoletazo de salida a la técnica paralela Multi-Swarm Optimization (Optimización Multi-Enjambre), siendo "*OptiFel: A Convergent Heterogeneous Particle Swarm Optimization Algorithm for Takagi-Sugeno Fuzzy Modeling*" [Cheung, N., 2013] el opus magnum en lo referido a dicha variante. Es reseñable que este enfoque multi-enjambre puede ser utilizado como un fiel aliado a la hora de lidiar con problemas adscritos a la Optimización Multi-Objetivo.
- PSO de *Hibridación (Hybridization PSO)*: Variante de ideología absolutamente abierta, cuya finalidad última es la de combinar el algoritmo PSO con todo tipo de técnicas, que fluctúan desde la Optimización basada en Biogeografía (*Biogeography-based Optimization*) hasta la incorporación de un método efectivo de aprendizaje en el propio algoritmo PSO, adicionando técnicas de rabiosa actualidad en el campo de la Inteligencia Artificial al mismo. La variante PSO de Hibridación pretende, en última instancia, mejorar el rendimiento de la técnica mediante la sinergia con otras metodologías.

Una vez descrito el funcionamiento del algoritmo PSO, su fundamento matemático, características y variaciones, destacan las distintas aplicaciones del mismo. Tal y como es costumbre en el ámbito de la Computación Natural, las aplicaciones factibles del algoritmo PSO están, prácticamente, acotadas por la imaginación. Entre las mismas, destacan:

- Optimización de la Potencia Reactiva y el Control de Voltaje: Una útil aplicación del algoritmo PSO se corresponde por el control de los parámetros eléctricos anteriormente citados por parte del Sistema de Alimentación Urbano nipón. En este caso, la técnica PSO se utiliza para esclarecer una estrategia de control que intercala variables de control tanto continuas como discretas desde el punto de vista matemático, desembocando en una suerte de versión híbrida del algoritmo PSO, que aúna valores binarios con valores reales. Es reseñable que la estabilidad en el voltaje se consigue, en este sistema, mediante una técnica de flujo controlado y continuado del mismo.
- Detección de Temblores adscritos a la enfermedad de Parkinson: El diagnóstico del temblor humano característico de la enfermedad de

Parkinson en una etapa temprana, es una acción tan difícil como crucial en el desarrollo de la enfermedad y el devenir del paciente. Por esta razón, el algoritmo PSO ha sido utilizado, en el ámbito de una Red Neuronal Artificial que contiene la información biométrica de los pacientes, con el objeto de evolucionar dicha red y distinguir aquellos sujetos sanos de aquellos que, desgraciadamente, sufren de los temblores característicos de la enfermedad.

- Evolución de Redes Neuronales Artificiales: Siguiendo en el ámbito de las redes comenzado con el anterior ejemplo de aplicación, el artículo "*Particle Swarm Optimization: Developments, Applications and Resources*" [Eberhart, R.C, 2010] pone en evidencia que el algoritmo PSO no sólo ha sido utilizado para evolucionar los aspectos más puramente relacionados con la transmisión en las redes, sino a su vez su estructura. Entendiendo la estructura de la Red Neuronal Artificial (RNA en lo sucesivo) como un problema y no como una topología, se pretende adoptar dicho problema desde un enfoque de optimización que persiga la optimización de la Propagación hacia Atrás de Errores (*Backpropagation*), vital en las RNA. Es reseñable que la técnica de la Propagación hacia Atrás de Errores consiste en un algoritmo de aprendizaje supervisado, que hace las veces de función de entrenamiento cognitivo en las RNA, por lo que la eficacia en la implementación de esta técnica será vital en el aprendizaje de la red en cuestión.
- Retirado de metal en procesos de fresado industrial: Dada la importancia asociada a la efectividad del fresado en los quehaceres industriales, se destaca que, a pesar de que el software de control ha mejorado con el paso de los años, este dista mucho aún de ser eficiente. Esta es la razón que motiva la emergencia de nuevas soluciones basadas en la brillante dupla RNA-PSO enfocada a una optimización de índole multi-dimensional, cuyos esfuerzos se están enfocando al Diseño Asistido por Computador (*Computer-Aided Design*) y a la Manufacturación Asistida por Ordenador (*Computer-Aided Manufacturing*).
- Optimización del Diseño de las Redes de Transporte (TNDP; *Transportation Network Design Problem*): El algoritmo PSO se aplica como metodología de selección entre un conjunto de alternativas, como pueden ser un nuevo conjunto de aristas que conectan nodos de la red, que minimizan un objetivo (típicamente el tiempo de transmisión entre pares), mientras mantienen el gasto de recursos entre unos límites establecidos *a priori*. La aplicación anteriormente citada se condensa brillantemente en el artículo "*Application of particle swarm optimization to transportation network design problem*" [Babazadeh, A., 2011], por lo que se referencia al interesado lector al mismo para ampliar información al respecto.

- Crecimiento controlado de cepas de producción de microorganismos: En esta aplicación concreta, los esfuerzos del algoritmo PSO se enfocan a la concepción de la mezcla de productos óptima para conseguir una cepa de producción de microorganismos concreta. La creación de tales microorganismos, que destacan por manufacturar elementos químicos de especial interés, tiene como pilar fundamental el algoritmo PSO, que es capaz de proveer los datos concretos referidos a la mezcla de productos que dobla la secreción de elementos químicos. Es digno de mención que el algoritmo PSO, dada su naturaleza, logra examinar un porcentaje mucho más amplio del espacio de búsqueda que las técnicas tradicionales, anteriormente aplicadas en tales entornos de producción de cepas de índole microorgánica. A título anecdótico, se presenta seguidamente una figura correspondiente a un conjunto de cepas de producción de microorganismos encargadas de optimizar el proceso de crecimiento del *Olea* *Europaea* (Olivo común):

Figura XXXIX: Cepas de producción de microorganismos de potenciación del crecimiento del *Olea* *Europaea*

- Estimador del estado de carga para vehículos eléctricos: El algoritmo PSO ha sido satisfactoriamente utilizado, de nuevo en conjunto con el algoritmo de Propagación de Errores hacia Atrás descrito anteriormente en el presente epígrafe, para aumentar el carácter cognoscitivo de las Redes Neuronales Artificiales como un estimador del estado de carga de los vehículos eléctricos, en boga actualmente. Se destaca que la determinación del estado de la batería en carga es un importante reto en el desarrollo de la tecnología vehicular eléctrica e híbrida.
- A modo de colofón en relación a las aplicaciones del algoritmo PSO, se destaca que dicho algoritmo, al igual que sus homólogos en el ámbito de la

computación natural, puede ser aplicado para resolver la mayor parte de problemas de optimización, así como problemas que pueden ser convertidos a problemas de optimización. Entre las numerosas áreas de aplicación de tan vigorosa técnica algorítmica, destaca el desarrollo de sistemas automático, la optimización multi-objetivo, la clasificación y reconocimiento de patrones, el modelado de sistemas biológicos, la simulación e identificación en los procesos de toma de decisiones a nivel robótico y la lógica difusa, entre un largo etcétera.

Englobando los conceptos anteriormente mostrados, característicos del algoritmo *Particle Swarm Optimization*, se presentará seguidamente un software que traslada la técnica al ámbito puramente computacional. De forma previa, cabe recordar al lector someramente que PSO constituye un método de optimización estocástico basado en una población de partículas. Dicho método, desarrollado por los doctores Eberhart y Kennedy en 1995, toma su inspiración del comportamiento gregario de las bandadas de aves, así como de los bancos de peces.

Es reseñable que el algoritmo PSO consta de numerosas similitudes con la técnica propia de la Computación Natural, extensamente descrita en el presente documento, conocida como Algorítmica Genética (GA). En ambas vertientes, el sistema se inicializa con una población (de individuos o partículas) que se corresponde con soluciones aleatorias, mientras que el algoritmo persigue una solución óptima mediante la actualización de iteraciones. No obstante, el algoritmo PSO carece de los operadores genéticos que impregnan los GA, pues en aquel las partículas (soluciones potenciales) se desplazan a través del espacio de búsqueda buscando el valor óptimo.

Como se relata en el presente epígrafe, cada partícula mantiene el estado de sus homólogas en el espacio de búsqueda asociado con la mejor solución encontrada hasta el momento. Tal solución se denomina *pBest*. Por otro lado, otro valor que podría ser óptimo hipotéticamente, es mantenido por PSO como el mejor valor obtenido hasta el momento por cualquier partícula en la población. Dicho valor será conocido como *lBest*. Cuando una partícula toma como referencia la población en su completitud, el mejor valor hallado se denomina óptimo global, *gBest*.

Por tanto, el concepto del algoritmo PSO se traduce en que, a cada iteración, las partículas sufren una aceleración que persigue su acercamiento a las localizaciones denominadas como *pBest* y *lBest*. Tal aceleración se computa en base a un valor aleatorio, generado dinámicamente en el propio algoritmo.

Seguidamente se procederá a la inclusión y explicación del código fuente adscrito al algoritmo PSO. Dicho *software*, implementado en el lenguaje de programación

Java (Version 1.8.0), puede obtenerse, al igual que el resto de software involucrado en la presente Tesis Doctoral, en el repositorio GitHub dedicado; <https://github.com/ClemencioMoralesLucas/PhD>. Con respecto al epígrafe que nos ocupa, el código fuente está disponible bajo la URI <https://github.com/ClemencioMoralesLucas/PhD/tree/master/src/phd/cml/particle/swarm/optimization>. Por motivos didácticos, se obviarán ciertas partes del código fuente dada su redundancia conceptual, por lo que se referencia al lector al repositorio *GitHub*. Código fuente de *Particle Swarm Optimization* para una visión completa del *software*.

Tal y como se ha descrito exhaustivamente a lo largo del presente epígrafe, el algoritmo PSO posibilita, entre otros aspectos, la posibilidad de resolver ecuaciones matemáticas multidimensionales entendidas como un problema matemático de optimización. En concreto, el *software* desarrollado tiene como objetivo la *minimización* de la siguiente fórmula bidimensional [Manalu, 2018]:

$$f(x,y) = (2.8125 - x + xy^4)^2 + (2.25 - x + xy^2)^2 + (1.5 - x + xy)^2$$

Donde se presupone que:

$$1 \leq x \leq 4 \quad y \quad -1 \leq y \leq 1$$

Como el atento lector intuirá, el *software* propuesto requerirá de la representación de ciertas magnitudes de índole física, tales como la *posición* y la *velocidad*, claves en el algoritmo PSO. Por tal motivo, procede la extrapolación de sus características en una clase *abstracta*, que sentará las bases necesarias para abordar la implementación de un mecanismo de herencia. Dicha clase, denominada *PhysicalMagnitude.java*, figura seguidamente:

```
package phd.cml.particle.swarm.optimization;

/**
 * Created by Clemencio Morales Lucas on 14/12/2015.
 */
public abstract class PhysicalMagnitude {

    private double[] values;

    public PhysicalMagnitude(final double[] values) {
        this.setValues(values);
    }

    public double[] getValues() {
        return this.values;
    }
}
```

```

        public void setValues(final double[] values) {
            this.values = values;
        }
    }
}

```

Tal y como puede apreciarse, la clase superior representa una variable de carácter físico, con su correspondiente valor, `getter` y `setter`. Gracias a dicha clase, la codificación de sus homólogas adscritas a la velocidad y a la posición se antoja trivial. En pos de evitar redundancias, se presenta a continuación la clase `Velocity.java`, simétrica a `Location.java`, que a su vez hereda de `PhysicalMagnitude.java`:

```

package phd.cml.particle.swarm.optimization;

/**
 * Created by Clemencio Morales Lucas on 14/12/2015.
 */
public class Velocity extends PhysicalMagnitude {

    public Velocity(final double[] values) {
        super(values);
    }
}

```

De forma idéntica a la previamente descrita algorítmica genética, el algoritmo PSO se apoya en una *población*, como ya se ha especificado. Dicha población, formada por *partículas*, es un aspecto clave en el algoritmo. Por esta razón, persiguiendo una granularidad fina en el *software*, se presenta la clase `Particle.java`, cuya explicación exhaustiva figurará seguidamente:

```

package phd.cml.particle.swarm.optimization;

/**
 * Created by Clemencio Morales Lucas on 14/12/2015.
 */
public class Particle {

    public static final double LOCATION_X_LOW = 1;
    public static final double LOCATION_X_HIGH = 4;

    public static final double LOCATION_Y_LOW = -1;
    public static final double LOCATION_Y_HIGH = 1;

    public static final double VELOCITY_LOW = -1;
    public static final double VELOCITY_HIGH = 1;

    private double fitnessValue;
    private PhysicalMagnitude velocity;
    private PhysicalMagnitude location;
}

```

```

    public Particle(final double fitnessValue, final PhysicalMagnitude
velocity, final PhysicalMagnitude location) {
        this.fitnessValue = fitnessValue;
        this.velocity = velocity;
        this.location = location;
    }

    public Particle(){}

    public PhysicalMagnitude getVelocity() {
        return this.velocity;
    }

    public void setVelocity(final PhysicalMagnitude velocity) {
        this.velocity = velocity;
    }

    public PhysicalMagnitude getLocation() {
        return this.location;
    }

    public void setLocation(final PhysicalMagnitude location) {
        this.location = location;
    }

    public double getFitnessValue() {
        fitnessValue = evaluate(location);
        return fitnessValue;
    }

    /*
    This evaluation function solves the following problem:
    Find an 'x' and 'y' value that minimize the following function:
    
$$f(x, y) = (2.8125 - x + x * y^4)^2 + (2.25 - x + x * y^2)^2 + (1.5 - x + x*y)^2$$

    Where  $1 \leq x \leq 4$ , and  $-1 \leq y \leq 1$ 
    */
    public static double evaluate(final PhysicalMagnitude location) {
        double result;
        final double x = location.getValues()[0];
        final double y = location.getValues()[1];

        result = Math.pow(2.8125 - x + x * Math.pow(y, 4), 2) +
            Math.pow(2.25 - x + x * Math.pow(y, 2), 2) +
            Math.pow(1.5 - x + x * y, 2);

        return result;
    }
}

```

La presente clase provee, entre otros, los siguientes aspectos fundamentales:

- Constantes adscritas a los límites inferiores y superiores de las dos variables involucradas en la ecuación a resolver.
- Constructor triparamétrico, que necesita de un valor de aptitud (*fitnessValue*) y dos magnitudes físicas (*velocity* y *location*).
- Método *evaluate* para la resolución de la ecuación propuesta. Tal y como puede apreciarse, dicha resolución se basa en la posición de cada partícula.

Seguidamente se presenta la importante clase *ParticleSwarmOptimization.java*, pilar fundamental del *software* en cuestión. Es reseñable que, dada la elevada longitud de dicha clase, se focalizará en los aspectos más trascendentes de la misma, referenciando al lector interesado en visualizar la clase en su completitud al repositorio de *GitHub* anteriormente citado. En concreto, se hará énfasis en el método que aglutina la algorítmica PSO, denominado *execute*:

```
package phd.cml.particle.swarm.optimization;

/**
 * Created by Clemencio Morales Lucas on 14/12/2015.
 */

import java.util.Vector;
import java.util.Random;

public class ParticleSwarmOptimization {

    public static final double ERR_TOLERANCE = 1E-20;
    // Rest of attributes...

    public void execute(final boolean verboseMode) {
        int iteration = 0;
        double w;
        double error = 9999;

        this.initializeSwarm();
        this.updateFitnessList();

        for(int i=0; i < SWARM_SIZE; i++) {
            pBest[i] = fitnessValueList[i];
            pBestLocation.add((Location) swarm.get(i).getLocation());
        }

        while(iteration < MAX_ITERATION && error > ERR_TOLERANCE) {
            // First action -> Update pBest
            for(int i=0; i < SWARM_SIZE; i++) {
                if(fitnessValueList[i] < pBest[i]) {
                    pBest[i] = fitnessValueList[i];
                    pBestLocation.set(i, (Location)
swarm.get(i).getLocation());
                }
            }
        }
    }
}
```

```

    }

    // Second action - Update gBest
    int bestParticleIndex = getMinimumPosition(fitnessValueList);
    if(iteration == 0 || fitnessValueList[bestParticleIndex] < gBest) {
        gBest = fitnessValueList[bestParticleIndex];
        gBestLocation = (Location)
    swarm.get(bestParticleIndex).getLocation();
    }

    w = W_UPPER_BOUND - (((double) iteration) / MAX_ITERATION) *
(W_UPPER_BOUND - W_LOWER_BOUND);

    for(int i=0; i<SWARM_SIZE; i++) {
        double r1 = random.nextDouble();
        double r2 = random.nextDouble();
        Particle p = swarm.get(i);

        // Third step - Update velocity
        double[] newVelocity = new double[PROBLEM_DIMENSION];

        newVelocity[0] = (w * p.getVelocity().getValues()[0]) +
            (r1 * C1) * (pBestLocation.get(i).getValues()[0] -
p.getLocation().getValues()[0]) +
            (r2 * C2) * (gBestLocation.getValues()[0] -
p.getLocation().getValues()[0]);

        newVelocity[1] = (w * p.getVelocity().getValues()[1]) +
            (r1 * C1) * (pBestLocation.get(i).getValues()[1] -
p.getLocation().getValues()[1]) +
            (r2 * C2) * (gBestLocation.getValues()[1] -
p.getLocation().getValues()[1]);

        Velocity vel = new Velocity(newVelocity);
        p.setVelocity(vel);

        // Fourth step - Update location
        double[] newLocation = new double[PROBLEM_DIMENSION];

        newLocation[0] = p.getLocation().getValues()[0] +
newVelocity[0];
        newLocation[1] = p.getLocation().getValues()[1] +
newVelocity[1];

        Location loc = new Location(newLocation);
        p.setLocation(loc);
    }

    //Observation: The function ought to be minimized, thus it means
    it's getting closer to 0
    error = Particle.evaluate(gBestLocation) - 0;

    if(verboseMode){
        printCurrentResults(iteration); }

    iteration++;
    updateFitnessList();
}

```

```
        printFinalResults(iteration-1);  
    }
```

Tal y como puede apreciarse, dicho método, que transita en un bucle cuya condición de término es un determinado número de iteraciones unido a un control de tolerancia al posible error presente en el cálculo matemático, inicializa (de manera aleatoria) la posición y la velocidad de las partículas presentes en el enjambre. Posteriormente, el método calcula la mejor posición particular conocida para cada partícula de la población, `pBest`. A continuación, la lista de valores de aptitud, `fitnessValueList`, se recorre con la finalidad de hallar el mejor valor `pBest` conocido hasta el momento. Seguidamente, de manera análoga se actualiza `gBest`, variable que albergará el mejor valor conocido por el enjambre en su conjunto. Llegado tal instante, las variables `velocity` y `location` son actualizadas para cada partícula, denotando el movimiento llevado a cabo por las partículas que componen el enjambre. Finalmente, se evalúa el resultado de la ecuación gracias al método estático `evaluate` contenido en la clase `Particle.java`, para, en caso de que el algoritmo opere en modo verboso, imprimir los resultados por pantalla, incrementar el número de iteración y actualizar la lista de funciones de aptitud.

Adviértase que, estableciendo un símil con una bandada de aves, las partículas en el algoritmo vuelan hacia la mejor solución de la ecuación propuesta, ajustando en cada iteración su posición y velocidad de bandada (enjambre) en pos de lograr tal objetivo, correspondiéndose el anteriormente citado esquema con la meta última de optimización que impregna el espectro del algoritmo PSO.

Finalizada la descripción de la clase anterior, vital en el *software* en cuestión, se presenta el siguiente código fuente, adscrito a la clase `Main.java` y suficiente para ejecutar el programa desarrollado. Adviértase que el *flag* `verboseMode` permite un cómodo control de la cantidad de trazas deseadas en el *software*:


```

package phd.cml.particle.swarm.optimization;

/**
 * Created by Clemencio Morales Lucas on 14/12/2015.
 */
public class Main {

    public static void main(String args[]) {
        final boolean verboseMode = true;
        new ParticleSwarmOptimization().execute(verboseMode);
    }
}

```

De forma previa a la inclusión de los resultados, es digno de mención que, con fines de claridad del código fuente y velando por el mayor estándar de calidad del *software* posible, se ha optado por no incluir numerosos comentarios de código fuente, ya que, en la humilde opinión del autor, el código claro y auto-explicativo *per se* es el mejor comentario posible.

A modo de colofón, se presenta el output correspondiente al *software* exhaustivamente descrito previamente:

```

Run Main
Iteration #97:
--> Best X value: 2.9999999970694637
--> Best Y value: 0.49999999898067266
--> Evaluation value: 4.760941208801523E-18

Iteration #98:
--> Best X value: 2.9999999976606575
--> Best Y value: 0.49999999928216327
--> Evaluation value: 2.3734437436193986E-18

Iteration #99:
--> Best X value: 2.9999999981522665
--> Best Y value: 0.49999999959339714
--> Evaluation value: 1.374794975821867E-18

-----
Solution found at Iteration #99. Values:
====> Best X: 2.9999999981522665
====> Best Y: 0.49999999959339714

Process finished with exit code 0

All files are up-to-date (moments ago)

```

Figura XL: Resultado de la ejecución del PSO representado en *ParticleSwarmOptimization.java*

Tal y como puede apreciarse en la captura de pantalla superior, el algoritmo resuelve la función, finalizando en la iteración noventa y nueve dada la invariancia presente. Como puede apreciarse, los resultados arrojados por el algoritmo se *estabilizan* notoriamente, presentando una muy ligera variación:

$$f(x,y) = (2.8125 - x + xy^4)^2 + (2.25 - x + xy^2)^2 + (1.5 - x + xy)^2$$

Determinando que la solución que minimiza $f(x,y)$ se corresponde con los siguientes valores:

$$x = 2.9999999981522665$$

$$y = 0.49999999959339714$$

A modo de colofón al presente epígrafe, se persigue realizar especial énfasis en la vasta expansión del algoritmo PSO, así como su evolución. Dicho algoritmo, que fue concebido como una mera curiosidad en sus orígenes, goza de una expansión sobresaliente, atrayendo el interés de numerosos científicos a lo largo y ancho del mundo.

En el año 2001, Eberhart y Kennedy, afirmaron que [nosotros] "*estamos mirando al algoritmo en su juventud, lleno de potencial y fértil, con nuevas ideas y perspectivas*". A pesar del paso de los años, aún quedan numerosas preguntas sin respuesta concernientes al algoritmo PSO, factor que pone de evidencia que, de la misma forma que Eberhart y Kennedy, la humanidad sigue mirando al algoritmo PSO en su juventud. La maduración de dicho algoritmo supone un prometedor y apasionante reto, en el que a buen seguro la comunidad científica internacional concentrará su esfuerzo en el futuro cercano.

2.1.5 Grammatical Evolution

En el seno de la irlandesa Universidad de Limerick (*University of Limerick*), fruto del excelente trabajo de los investigadores Conor Ryan, Michael O'Neill y J.J. Collins, surge, en el año 1998, el paradigma denominado *Grammatical Evolution* (Evolución Gramática) [Ryan et al., 1998]. Tal paradigma parte de la capacidad del Algoritmo Genético (GA, exhaustivamente descrito en el epígrafe 2.1.2 *Genetic Algorithms* del presente documento) de *evolucionar software*, escrito en cualquier lenguaje de programación, utilizando una cadena de caracteres de longitud variable. Por esta razón, el algoritmo GE puede interpretarse como una variación de su homólogo GA, que, aunque conste de menor enjundia que otras técnicas adscritas a la Computación Natural, bien merece el epígrafe que nos ocupa, como se demostrará en el mismo.

En la casuística concreta de la Evolución Gramática, (Grammatical Evolution; GE en lo sucesivo) el genoma, codificado en formato binario, determina qué reglas de producción serán usadas en el proceso de transición genotipo-fenotipo-*software*. Es reseñable que dichas reglas de producción estarán representadas, típicamente, en la notación BNF (*Backus-Naur Form*). Se recuerda al lector que la notación BNF se corresponde con un *metalenguaje* (lenguaje que representa otro lenguaje a su vez) cuyo objetivo es el de representar gramáticas libres de contexto, o lo que es lo mismo, *lenguajes formales*. Dicha notación define un sistema de reglas de derivación para lenguajes, en el que, típicamente un símbolo se representa en base a la siguiente fórmula:

$$< \textit{simbolo} > ::= < \textit{expresión de símbolos} >$$

En dicha fórmula, la sección izquierda, *<simbolo>*, se corresponde con un carácter *No Terminal*, mientras que la parte opuesta, *<expresión de símbolos>*, representa una secuencia de símbolos separados por el carácter | o barra vertical, que indica la opcionalidad. Los caracteres que siempre aparecen en el lado derecho se denominan *Terminales*. A título anecdótico a la par que explicativo, se presenta la siguiente figura, correspondiente a las directrices *BNF* que regulan la denominación de los identificadores de variable en Java:

$$\begin{aligned}
\langle \text{identifier} \rangle &::= \langle \text{initial} \rangle \mid \langle \text{initial} \rangle \langle \text{more} \rangle \\
\langle \text{initial} \rangle &::= \langle \text{letter} \rangle \mid _ \mid \$ \\
\langle \text{more} \rangle &::= \langle \text{final} \rangle \mid \langle \text{more} \rangle \langle \text{final} \rangle \\
\langle \text{final} \rangle &::= \langle \text{initial} \rangle \mid \langle \text{digit} \rangle \\
\langle \text{letter} \rangle &::= a \mid b \mid c \mid \dots z \mid A \mid B \mid \dots \mid Z \\
\langle \text{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}$$

Figura XLI: Directrices BNF para los identificadores de variable en Java

Es importante destacar, a modo de inciso, que un lenguaje formal se corresponde con un conjunto de símbolos primitivos y reglas [Mellema, 2008]. Estas reglas servirán para unir dichos símbolos, que a su vez estarán especificados de manera *formal*. El conjunto de símbolos primitivos se denomina *alfabeto* del lenguaje, mientras que las reglas denotan la *gramática formal* (también llamada *sintaxis*). Por otro lado, una cadena de símbolos construida en base a la gramática se denomina *palabra* del lenguaje. De hecho, un lenguaje formal es equivalente al conjunto de todas sus palabras. Al contrario de lo que ocurre con el alfabeto y con cada palabra bien formada, ambos finitos por definición, un lenguaje formal puede estar formado por un número infinito palabras bien formadas.

Retomando la idea de la técnica GE, ésta extrapola la lógica de los GA, teniendo por objetivo el de encontrar una instancia del lenguaje generado por la gramática. Dicha instancia se corresponderá con un *software* ejecutable que logre un aceptable valor de *fitness* para una función objetivo dada. De forma clásica, en la programación genética se manipula una expresión estructurada en forma de árbol, mientras que, como se explicará a lo largo del presente epígrafe, en el caso de la GE esta manipulación se lleva a cabo sobre una cadena numérica. Posteriormente, esta cadena se transforma en un programa mediante la aplicación directa de la gramática. Como el atento lector habrá vislumbrado, uno de los beneficios de la técnica GE frente a la GA es la simplificación en el proceso de *mapping*, que minimiza la dificultad de aplicar la búsqueda a distintos lenguajes de programación.

Por tanto, el algoritmo GE se concibe con una aproximación a la Programación Genética, que provee la generación de *software* en un lenguaje de programación arbitrario. Adviértase que GE pretende explotar una rica modularidad en su diseño, resultando en un sistema altamente flexible, potente y sencillo de utilizar. Es reseñable que la técnica GE debe su diseño, y, por consiguiente, su potencia, a la inspiración que le brindan los sistemas biológicos presentes en la naturaleza.

En cuanto al origen de la algorítmica adscrita a la técnica GE, es fundamental tener presente que esta es una rama de la Programación Genética, que difiere, en cierto modo, de la misma en tres aspectos atómicos: realiza un proceso de *mapping* ontogenético (desde el periodo *embrionario* en una percepción computacional) desde el genotipo al fenotipo o, en este caso, programa, utiliza una gramática para establecer las estructuras legales en el espacio fenotípico y utiliza genomas de corte lineal. Debe advertirse que los genotipos son listas ordenadas de números enteros que se codifican para seleccionar las reglas de la gramática de contexto libre proporcionada. El fenotipo, por su parte, se corresponde con una estructura arbórea que se evalúa recursivamente.

Habiendo horadado la superficie del algoritmo, el ávido lector puede ser asaltado por un interrogante, y es que, ¿cuál es el pilar fundamental de la algorítmica GE? Una vez más, el *motor* de evolución algorítmica descrito por John Holland en 1975, inspirado por los *opus magnum* de Spencer y Darwin, emerge como la solución a dicha cuestión: En el ámbito del GE, se mantiene una población de soluciones (genotipo), en la que cada individuo será evaluado a fin de establecer su rendimiento con respecto al programa abordado. Dado que el conjunto genotípico en el ámbito GE está formado por programas, el rendimiento de cada uno de dichos elementos será conocido, una vez más, por el término *aptitud*, o bien su homólogo anglosajón *fitness*. Así, aquellos individuos que presenten un rendimiento superior en las tareas asignadas, recibirán un mayor valor de aptitud, aumentando por ende su probabilidad de supervivencia de cara a la siguiente etapa reproductiva. Como consecuencia directa a la premisa anterior, estos individuos estarán más cerca de conseguir uno de los fines últimos de la genética; Transferir su material genético a la siguiente generación. Se recuerda que los conceptos anteriormente descritos toman su base en la selección natural, en la que aquellos individuos mejor adaptados a su entorno, o, en terminología propia de la Computación Evolutiva, aquellos individuos mejor adaptados al problema, sobreviven para producir la siguiente generación. Es obligado referenciar al epígrafe 2.1.2 *Genetic Algorithms* para una información más exhaustiva al respecto. Adviértase que la inspiración provista por la naturaleza al campo computacional que nos ocupa no finaliza en la descripción de procesos biológicos básicos o términos adscritos a la naturaleza, sino que consta de una potencia directamente proporcional al nivel de profundidad con el que se realice la inmersión en los procesos de índole natural. La extrapolación de estos procesos al ámbito informático constituye una prometedora fuente de conocimientos de cara al futuro próximo, procesos en los que destaca, en especial, el campo de la Biología Molecular.

Dicho ámbito, de actualidad cuasi-rabiosa, ha atraído numerosos científicos hacia el funcionamiento interno de la unidad atómica de todo ser vivo, la célula. En dicho ámbito, destacan los descubrimientos realizados en el campo del ADN

hereditario, el descubrimiento de nuevas clases de proteínas así como de sus unidades estructurales, el área más íntima del código genético y la compleja red de señales de aviso existente entre las distintas células que componen un organismo. Es destacable, entre otros, el conocido como el *Dogma Central de la Biología Molecular*, que describe el proceso de traducción y transcripción que acontece entre, por ejemplo, el ADN y el ARN, que posteriormente desembocará en la creación de proteínas. Las proteínas son responsables, entre otros, en la generación de los rasgos fenotípicos que distinguen al individuo entre sus homólogos, tales como el color del iris y la altura. En términos de la Computación Evolutiva, estos términos desembocan en la diferencia entre genotipo y fenotipo. La separación del espacio de búsqueda y el espacio de soluciones no debe considerarse asunto baladí, pues consigue la sencilla adopción de un proceso de *mapping* genotipo-fenotipo, en el ámbito que nos ocupa, a través de una gramática o lenguaje formal. Como será descrito a lo largo del presente epígrafe, tal característica permitirá disfrutar a la técnica GE de beneficiosas ventajas, entre las que sobresale una idea principal: la habilidad de representar estructuras genotípicas de una manera más abstracta a través del apoyo brindado por las gramáticas [Alfonseca et al., 2015].

Es lícito afirmar que la técnica GE consta de características únicas en comparación con otros algoritmos de corte evolutivo, entre las que destaca la degeneración de código genético que facilita la aparición de mutaciones neutrales, en las que varios genotipos pueden representar, tras ejecutar el algoritmo, en el mismo fenotipo, así como el encapsulado del genotipo durante el proceso de *mapping*. Dicho proceso permite la reutilización del mismo genotipo para la generación de distintos fenotipos.

Tal y como se ha evidenciado, la técnica GE toma su inspiración a partir de la naturaleza. De esta manera, esta metodología esboza los principios mínimos necesarios que permiten una representación extrapolada, de naturaleza abstracta, del problema a solucionar. Esta abstracción constituye una portentosa fortaleza de la que el algoritmo se beneficia, pues permite, entre otros factores, los siguientes:

- Evolución de *software* en cualquier lenguaje de programación.
- Encapsulado que permite reutilizar áreas del *software* como si de material genético se tratara.
- Separación del espacio de búsqueda y del espacio de soluciones.

Una vez introducida la técnica GE, se está en condiciones de describir formalmente el algoritmo. Se recuerda que, aunque la metodología GE sea una de las variantes de mayor envidia del algoritmo GA (y en cierto modo de los

algoritmos evolutivos en términos generales), no presenta una complejidad hiperbólica. A saber, en la Evolución Gramatical:

1. Se define una gramática para describir la sintaxis del lenguaje.
2. Por otro lado, se concibe una función de coste para evaluar la calidad, en términos de su función de aptitud, de un *software*.
3. Se utiliza un algoritmo evolutivo, típicamente GA, para llevar a cabo una búsqueda en el espacio definido por todos los programas susceptibles de ser descritos por la gramática, persiguiendo la finalidad última de encontrar el *software* de menor coste posible. (En otras palabras, se busca hallar el programa que mejor función de aptitud o *fitness* presente.)

Es reseñable que, de forma previa a la evaluación de cada individuo, acontecen las siguientes acciones:

- El genotipo es utilizado para llevar a cabo el proceso de *mapping* del símbolo que convierte la definición gramatical, en formato BNF, en caracteres terminales. La gramática se utiliza en este punto para especificar el espectro de genotipos legales. Adviértase que el genotipo es representado como un *string* binario de longitud variable.
- El algoritmo GE realiza lecturas de *codones* (en terminología genética, secuencia de tres nucleótidos que conforman una unidad de código genético en una molécula de ADN o ARN) de un byte, mientras que el valor entero correspondiente a la secuencia de *codón* en bits se utiliza para determinar qué regla gramatical se utiliza cada vez que un carácter No Terminal a ser traducido presenta alternativas.
- La imagen adscrita a la regla de producción gramatical se concibe en base a la fórmula:

$$\textit{Imagen Terminal} = \textit{codón} \bmod \textit{Imágenes Terminales}$$

Donde *codón* es un valor entero e Imagen Terminal se corresponde con el número de *traducciones* disponibles para el carácter No Terminal objeto de estudio.

Por tanto, la técnica GE realiza un proceso de evolución de soluciones en pos de hallar una gramática especificada por el usuario. Por consiguiente, el espacio de búsqueda se minimiza notoriamente, factor que reduce a su vez la carga computacional del algoritmo. Es reseñable que el paradigma GE fue concebido a

tenor de la necesidad de separar el genotipo del fenotipo en el GA, pues en esta última técnica las entidades sobre las que el algoritmo opera y el *input* de la función de aptitud o *fitness* son las mismas. Adviértase que, de forma opuesta, en la metodología GE, el genotipo se corresponde con una lista ordenada de valores enteros que se codifican en pos de la selección de directrices asociadas al lenguaje formal seleccionado. No obstante, el fenotipo sigue el mismo patrón que el presente en los GA, es decir, una estructura de tipo arbórea que es recursivamente evaluada. Tal modelo se corresponde de manera fiel al mecanismo genético que opera en la mayoría de procesos naturales, en los que existe una separación explícita entre el genotipo orgánico y la representación final del fenotipo en proteínas. La siguiente figura ilustra fielmente las similitudes existentes en dicha extrapolación:

Figura XLII: Analogías en el proceso evolutivo entre la GE y un sistema biológico clásico

Es reseñable, por otro lado, que en el ámbito de la técnica GE es posible llevar a cabo la búsqueda mediante métodos tales como PSO, objeto de investigación del anterior epígrafe del presente documento. Tan beneficiosa sinergia se posibilita de la mano de la naturaleza modular de la metodología GE, que permite intercalarla con un amplio elenco de algoritmos de búsqueda [de Mingo et al., 2009].

Tal y como se ha especificado a lo largo del epígrafe actual, una poderosa ventaja de la técnica GE es el uso de gramáticas o lenguajes formales en la misma, pero, ¿cuál es exactamente el impacto positivo que tal uso provee? La respuesta a dicha pregunta se bifurca en dos grandes vertientes benignas, cuya descripción figura seguidamente:

- Generalidad (*Generality*): Tal y como se especifica en el epígrafe homónimo del presente documento, una de las ventajas de los GA es la ínfima cantidad de información necesaria sobre el problema a acometer.

De esta forma, los GA son susceptibles de aplicación incluso en casuísticas en las que el proceso sujeto a análisis presenta aspectos borrosos y poco claros. No obstante, aunque la cantidad de información disponible sobre el problema es directamente proporcional a su eficiencia, en la adición de información al *software* requiere, de forma casi obligada, la modificación del código fuente. En contraposición, en el caso de la GE, que deriva del paradigma de los GA, esta modificación puede realizarse sobre la gramática en detrimento del código fuente, por lo que el código fuente se beneficia de un incremento de potencia sin necesidad de sacrificar la legibilidad e inmutabilidad del código. Tal y como O'Neill y Ryan afirman en el 2003, "*The more one knows about a problem, the more one can enrich the grammar with all manner of insider information about the problem.*" ("Cuanto más información se tenga sobre un problema, más puede enriquecerse la gramática en función de la información adicional sobre el problema".) Así, el resultado arrojado tras la aplicación de un algoritmo adscrito a la GE puede estar representado en cualquier formato que pueda ser codificado por una gramática.

- Legalidad (*Legality*): Esta característica se corresponde con una propiedad adscrita a aquellas estructuras en las que una gramática correctamente formada siempre producirá como salida un conjunto válido. (Adviértase que este escenario obvia el caso en el que el código fuente entra en un bucle infinito, cae en una incongruencia, y situaciones análogas.) Este factor aparece debido a que, en vez de llevar a cabo la construcción de una estructura directamente, tal y como ocurre en los GA, dicha estructura es creada desde su base, a través de la aplicación iterada de una secuencia de derivación de la gramática, o lo que es lo mismo, una secuencia de elecciones que establece las reglas gramaticales a utilizar por parte del algoritmo GE.

A pesar de las anteriormente citadas ventajas, de elevada potencia como se razona en las líneas superiores, existen escenarios en los que esta robustez no es suficiente. Precisamente, con el fin de lidiar con las casuísticas más complejas, se presenta la *optimización evolutiva*, la cual se corresponde con un subconjunto de técnicas inspiradas por la evolución natural que se usan en casos donde:

- La solución a un cierto problema puede ser representada por una estructura muy concreta. A modo de ejemplo, supóngase que la solución es un *array* o vector de números binarios.
- Existe una función de coste que puede devolver con presteza el valor de aptitud o *fitness* de cada solución candidata.

- Solucionar el problema mediante el uso de técnicas de gradiente descendente es difícil o directamente imposible, pues la función de aptitud o coste tiene múltiples óptimos locales, o es, simplemente, discreta. Un famoso ejemplo de una función de coste discreta se corresponde con el Problema del Viajante (*Travelling Salesman Problem*).

A pesar de las posibles optimizaciones susceptibles de ser aplicadas en el algoritmo GE, se recuerda que la naturaleza estocástica de su homólogo GA no garantiza el hallazgo de una solución óptima. Además, la mayoría de problemas prácticos involucran espacios de búsqueda notoriamente amplios, por lo que la exploración del espacio de búsqueda en su completitud no es siempre computacionalmente viable.

Por tanto, ¿cómo se optimiza realmente una gramática a través de un mecanismo evolutivo? Para resolver dicha pregunta, debe recordarse que el algoritmo GA optimiza, únicamente, vectores numéricos. Mediante el proceso de *mapping* desde un vector de valores enteros a un programa usando una gramática, el algoritmo GA puede ser aplicado satisfactoriamente para evolucionar *software* mediante la siguiente secuencia:

1. La solución se representa como un *array* de valores enteros.
2. Se lleva a cabo una operación de *mapping* sobre el *array*, consiguiendo como resultado un programa a través de la gramática.
3. La función de coste mide la aptitud del *software* generado.
4. Se aplica cualquier técnica de optimización evolutiva en el *array* de enteros.

Habiendo descrito los aspectos referidos a las optimizaciones del algoritmo GE, bien merece la pena detenerse un instante en las variaciones del mismo. A pesar de la rabiosa actualidad de la algorítmica GE, existen versiones mejoradas y poderosas variaciones del algoritmo en la actualidad. La experimentación en base al algoritmo PSO para llevar a cabo la búsqueda en detrimento de la algorítmica genética supone, dado el sobresaliente rendimiento de sus resultados, una digna alternativa por méritos propios. Destaca que la bondad de dicho rendimiento es tan alta que esta sinergia es conocida como *Grammatical Swarm* (Enjambre Gramático), que da nombre al siguiente epígrafe del presente documento dada su enjundia. No obstante, a modo de avance, es destacable que el uso de dicha técnica ha supuesto, utilizando únicamente el modelo algorítmico de PSO clásico, la demostración de un interesante argumento: el algoritmo PSO es igualmente capaz de llevar a cabo el proceso de búsqueda en el GE esgrimiendo el mismo rendimiento que los GA. Adviértase la importancia hiperbólica de este resultado,

pues se recuerda que PSO es una técnica de *punto flotante*, por lo que, desde el punto de vista teórico, puede concebirse como computacionalmente más compleja. No obstante, esta complejidad puede ser reducida mediante la transición a un valor discreto de los valores involucrados en el algoritmo para su uso en el ámbito del paradigma GE.

Por otro lado, otra posible variación abordada con respecto a la técnica GE es la de la codificación de la información semántica adscrita a la gramática en pos de sesgar el proceso de búsqueda. Cabe destacar que esta aproximación se ha realizado únicamente desde el punto de vista teórico, por lo que su aplicación efectiva está aún en entredicho.

En cuanto a las posibles aplicaciones de la técnica GE se refiere, destaca que prácticamente cualquier aplicación que necesite un programa describible formalmente en base a una gramática, es susceptible de ser abordado mediante la técnica GE. El uso de una gramática permite la integración del conocimiento del dominio de la problemática y de una sintaxis del *software* personalizada, factor que añade una importante flexibilidad y precisión al algoritmo GE. Entre las aplicaciones existentes, destacan las obras de O'Neill y Brabazon, que han logrado aplicar, de forma satisfactoria, el algoritmo GE a la predicción de la bancarrota empresarial, la previsión del estado futuro de índices bursátiles, las calificaciones del crédito de bonos y un sinnúmero de aplicaciones de corte financiero. Por otro lado, la técnica GE ha sido utilizada, con un rendimiento sobresaliente, en simulaciones del modelo clásico de *Depredador-Presa* con el fin de explorar parámetros tales como la eficiencia por parte del primero, el *número de nicho* de presas disponibles y diversas mutaciones aleatorias que atañen la estabilidad ecológica y del medio ambiente.

Realizando un enfoque orientado hacia el aspecto más puramente práctico de la técnica algorítmica GE, destaca que existen numerosas bibliotecas adscritas al mismo, disponibles a su vez para un aceptable elenco de lenguajes de programación. Entre ellas, destacan:

- *GEVA (Grammatical Evolution in Java; Evolución Gramática en Java)* [GEVA, 2014]: Desarrollado por el equipo NCRA (*Natural Computing Research & Applications Group; Grupo de Investigación en Computación Natural & Aplicaciones*) de la Universidad de Dublín, la biblioteca actual es posiblemente una de las más populares en el ámbito de la algorítmica GE, por lo que es la seleccionada para describir en el presente epígrafe. Dicha descripción figurará seguidamente [Grammatical, 2018].
- *jGE*: Desarrollado por la Universidad de Bangor, pionera en el desarrollo de un *framework* para el uso de la algorítmica GE a fin de *evolucionar software Java*, *jGE* goza de una aceptable popularidad en la comunidad

científica internacional. Parte de su potencia puede estudiarse en [Si et al., 2018].

- *GERET*: Concebido e implementado por Pavel Suchmann en el año 2009, *GERET* constituye una potente biblioteca *Ruby* para la investigación en el ámbito de la GE. Ofreciendo la posibilidad de llevar a cabo una optimización multi-objetivo, la configuración *amigable* (*user-friendly*) en base a ficheros *YAML* y numerosos operadores genéticos así como algoritmos de búsqueda, *GERET* se posiciona como una poderosa alternativa para los programadores de *Ruby*.
- *libGE*: Esta biblioteca en C++ concebida por la irlandesa Universidad de Limerick, constituye una biblioteca que implementa el proceso de *mapping* adscrito a la técnica GE. Típicamente, *libGE* es utilizado por un algoritmo GE, que provee el *mapping* determinado desde una estructura genotípica hasta una fenotípica. En su implementación por defecto, lleva a cabo el *mapping* sobre un *string*, que dará como *output* un programa correcto desde el punto de vista sintáctico, y cuyo lenguaje está especificado de forma previa en una gramática BNF libre de contexto, o lo que es lo mismo, un lenguaje formal.
- *Ponyge*: De una entidad y enjundia mucho menor que *GEVA*, pues aquel se corresponde con una implementación en un único fichero de código, al contrario que *GEVA*, *Ponyge* es el nombre de la implementación en *Python* por parte de la universidad anteriormente citada para lidiar con el algoritmo GE [Bath, 2018]. Un enfoque similar es reseñable en [Cran, 2018].
- *DRP* (*Directed Ruby Programming*; Programación Ruby Direccionada): Desarrollada por Christophe McKeon, la técnica *Directed Programming* (Programación Dirigida, PD en lo sucesivo) consiste en una generalización de la algorítmica GE, permitiendo no sólo ejecutar el algoritmo GE, sino a su vez hacer lo propio con algoritmos genéticos escritos totalmente en el lenguaje de programación *Ruby* sin generar explícitamente un *string* como *output*. PD permite incluso concebir sistemas híbridos entre la programación genética y la técnica GE, donde una parte del sub-árbol GE es calculado mediante la aplicación de operadores y funciones escritas en *Ruby* mediante algoritmos genéticos. Es digno de mención que la biblioteca *DRP* constituye la primera implementación disponible de *DP* escrita en *Ruby puro*, sin lenguajes de programación de apoyo añadidos.

Descrito el algoritmo GE, así como sus variaciones, optimizaciones y bibliotecas disponibles, se procederá a la explicación exhaustiva de la biblioteca *GEVA*, de mayor popularidad entre las disponibles. En primer lugar, dicho *software* puede descargarse desde la URI <http://ncra.ucd.ie/geva/GEVA-v2.0.zip>. Una vez

obtenida la biblioteca, la misma ha de descomprimirse, obteniendo el directorio *GEVA-v2.0*. Es reseñable que, dado que la presente biblioteca está escrita en el lenguaje de programación Java, una versión de dicho lenguaje superior a V.1.5 debe estar instalada en el sistema anfitrión. En el directorio creado aparecerán los siguientes elementos:

- *bin*: Contiene los archivos ejecutables *GEVA.jar* y *GUI.jar*, así como las bibliotecas asociadas.
- *build.xml*: Fichero utilizado por ant para reconstruir la distribución completa tras cualquier modificación de un fichero de código fuente.
- *COPYING*: Alberga la descripción de la tercera versión de la licencia *GNU GPL*.
- *GEVA*: Contiene el código fuente de la aplicación, la documentación asociada al mismo y los ficheros *build* para *GEVA*.
- *GEVA_GUI.jar*: Punto de entrada principal de la aplicación. Permite ejecutar la GUI asociada a *GEVA*, invocando de forma implícita a *GUI.jar*, contenido en la carpeta *bin*.
- *GUI*: Alberga el código fuente, la documentación del mismo y los distintos ficheros adscritos a la *GUI* del sistema.
- *gui.config*: Fichero de utilidad usado para la configuración de la *GUI*, de tal forma que conoce la ubicación de *GEVA.jar*, así como la de los ficheros de propiedades y gramática.
- *LICENSE*: Contiene la información de copyright de *GEVA*, así como la información de la licencia de la distribución.
- *param*: Escindido en dos directorios; *Grammar*, que contiene los archivos de extensión *bnf* que contienen la gramática utilizada, así como las propiedades aplicadas a los mismos bajo el directorio *Parameters*. Los ficheros *ff.config* y *graph.config* contienen, a su vez, la configuración por defecto de la interfaz gráfica de usuario.

Con el fin de ejecutar la GUI (*Graphical User Interface*; Interfaz Gráfica de Usuario) de *GEVA*, ha de lanzarse el archivo *GEVA_GUI.jar*. Si el compilador Java existente en el sistema satisface el requisito citado previamente, la siguiente imagen aparecerá mientras se lleva a cabo la carga de la biblioteca:

Figura XLIII: Screenshot de carga correcta de la biblioteca GEVA v2.0 para algorítmica GE

Tras un breve instante, la aparición de la GUI de *GEVA* indicará tácitamente que la ejecución ha sido correcta mostrándose en pantalla:

Figura XLIV: Screenshot de la GUI asociada al software GEVA v2.0

Tal y como puede apreciarse en la captura de pantalla anteriormente presentada, la biblioteca consta de una serie de casos de prueba seleccionables en el *drop-*

down Properties. Por otro lado, la pestaña *GEVA* contiene cinco grandes grupos de opciones, a saber:

1. *General*: Permite ajustar los aspectos básicos del algoritmo GE, entre los que aparecen el comportamiento a adoptar por la librería cuando el programa finalice, el tipo de codón a utilizar, las generaciones a producir por el algoritmo GE, el tamaño de la población, el tamaño de *wrapping* a respetar y, opcionalmente, el archivo donde se desea que *GEVA* vuelque los resultados obtenidos.
2. *Fitness Function* (Función de Aptitud): Se escinde en dos opciones, *Fitness Function*, que permite seleccionar la Función de Aptitud a utilizar y *Grammar*, que define la ruta al fichero contenedor del lenguaje formal a usar.
3. *Replacement and Selection* (Reemplazamiento y Selección): Comenzando el bloque de opciones avanzadas asociadas a la biblioteca, otorga la opción de seleccionar el tipo de reemplazo a utilizar, el tamaño de *elite* del algoritmo GE, el operador de selección y la cardinalidad del mismo.
4. *Initialisation* (Inicialización): Permite configurar los aspectos asociados a la característica que le da nombre, así como la probabilidad de crecimiento de la población y su profundidad, en términos de descendencia, correspondiente.
5. *Other* (Otro): Define las características puramente genéticas involucradas en el algoritmo GE, a saber: Probabilidad de Cruzamiento, Punto de Cruzamiento y Probabilidad de Mutación.

Finalmente en cuanto a las opciones que la GUI ofrece, existen los botones *Run* (para ejecutar el algoritmo seleccionado con los parámetros introducidos), *Configure* (realización de ajustes principalmente relacionadas con las rutas del sistema), *Close* (cierre de la aplicación), *Save*, *New* y *Delete*, con el mismo comportamiento que la lógica apunta.

Una vez descrita atómicamente la GUI perteneciente a *GEVA*, y con el fin de ilustrar la potencia de dicha biblioteca, se tomará como ejemplo práctico el problema *HelloWorld*.

En este problema, el objetivo fundamental es el de encontrar la cadena de caracteres "*geva*", que puede estar contenida en un *string*. La función de aptitud, por ende, consiste en un simple recuento del número de caracteres de la solución candidata que satisfaga el *string* objetivo. Con el fin de lograr dicho resultado, se presenta la gramática utilizada, que puede encontrarse en la ruta *GEVA-*

v2.0\param\Grammar y cuyo nombre se corresponde con HelloWorld_grammar.bnf:

```
<prog> ::= <func> | <func><prog>
<func> ::= writeChar( <letter> );
<letter> ::= <vowel> | <consonant> | <space>
<space> ::= '_'
<vowel> ::= 'a' | 'o' | 'u' | 'e' | 'i'
<consonant> ::=
'q' | 'w' | 'r' | 't' | 'y' | 'p' | 's' | 'd' | 'f' | 'g' | 'h' | 'j' | 'k' | 'l' | 'z' | 'x' | 'c' | 'v' |
'b' | 'n' | 'm'
```

Tal y como puede apreciarse, un programa (`prog`) se define recursivamente mediante una función (`func`) o bien una función y un programa. Dicha función estará a su vez descrita por la escritura de una letra (`letter`), que puede ser un espacio (`space`), una vocal (`vowel`) o una consonante (`consonant`). La gramática anteriormente presentada generará `prog` aleatorios hasta que el *software* halle el *string* buscado, que, como se especificó anteriormente, se corresponde con los caracteres "geva" y puede constituir un subconjunto de un *string* de mayor cardinalidad.

Dado que la búsqueda se lleva a cabo siguiendo un criterio de minimización, los menores valores de aptitud son considerados mejores, por lo que la cuenta de caracteres se realiza sustrayendo el número de caracteres perseguido (n) del *string* objetivo. Tal criterio se refleja en la siguiente fórmula:

$$\text{Aptitud} = \text{LongitudStringObjetivo} - \sum_{i=1}^n (\text{caracterObjetivo}[i] - \text{caracterPrevisto}[i])$$

Una vez esgrimidos los argumentos anteriores, el *software* HelloWorld.properties puede ejecutarse presionando el botón Run en la GUI de GEVA, acción que provocará la aparición de la siguiente pantalla de resultados asociados a la ejecución del algoritmo GE:

Figura XLV: Screenshot asociado a la ejecución de HelloWorld.properties en GEVA v2.0

Como puede apreciarse en la figura superior, el *software* encuentra el sub-string "geva" en poco más de 66 milisegundos, por lo que, tal y como está establecido en su configuración, detiene la ejecución. La jugosa gráfica que aparece en la figura invita al análisis matemático de los resultados, presentando numerosos valores adscritos al mejor valor de aptitud (Best), el valor medio (Average) y a la varianza existente (Variance). Es reseñable que la ventana de resultados contiene numerosos datos de interés, tales como el tiempo de ejecución, el número de iteraciones y los elementos inválidos adscritos al genotipo, que pueden obtenerse de las pestañas Codon, Invalids y Other. Es digno de mención que los colores de las variables que figuran en la gráfica pueden modificarse a elección del usuario con el fin de lograr una óptima visualización.

Por otro lado en cuanto a la visualización de resultados en GEVA se refiere, es reseñable especificar el contenido de la pestaña Console, en la que puede apreciarse el conjunto de variables que intervienen en el algoritmo, así como el fenotipo logrado, el tiempo total de ejecución y el número de iteraciones aplicado:

Figura XLVI: *Screenshot* asociado a la sección Console de HelloWorld.properties en GEVA v2.0

Finalmente, una vez descrito el potente uso de GEVA y sus características, es lícito inferir a modo de conclusión que la algorítmica GE constituye un flexible, a la vez que potente, marco de generación automática de *software*. La sintaxis y la estructura de los programas son descritos, como se ha argumentado a lo largo del presente epígrafe, en función de una gramática libre de contexto, o lo que es lo mismo, un lenguaje de tipo formal. Su objetivo, será determinado por una función de evaluación o *fitness*, siendo esta la marca característica de la mayoría de algoritmos derivados de la Computación Evolutiva, como es el caso de la técnica GE, donde una búsqueda evolutiva se lleva a cabo sobre la gramática utilizada a fin de encontrar el *software* que minimice la función de aptitud. Tal y como a buen seguro el ávido lector habrá extraído del sucinto pero concentrado epígrafe actual, la investigación sobre la algorítmica GE supone un reto a la vez que un apasionante ámbito científico contemporáneo.

2.1.6 Grammatical Swarm

A lo largo del presente epígrafe 2.1 *Paradigmas de Computación involucrados*, se ha evidenciado como el concepto de *enjambre* permanece, explícita o implícitamente, de forma evidente o soterrada, impregnando la idiosincrasia del Paradigma de la Computación Natural. Como se describirá a lo largo de la presente sección, esta es, precisamente, la razón de ser que motiva la existencia del Paradigma *Grammatical Swarm* (Enjambre Gramático, GS en lo sucesivo), que, como se especificará seguidamente, surge como una sinergia entre la técnica *Grammatical Evolution* (GE) y su homóloga *Particle Swarm Optimization* (PSO).

Durante las últimas décadas, numerosos investigadores se han visto atraídos hacia el estudio de los pormenores adscritos al concepto de *enjambre*, dada su importancia ya innecesaria de recordar. Tras el estudio detenido de una bandada de aves o un banco de seres acuáticos, se evidencia que dichos *enjambres* presentan, como características atómicas, la robustez y flexibilidad, así como una auto-organización sobresaliente, casi precognitiva, y una sublime resiliencia a la hora de lidiar con grupos de depredadores. De forma voluntaria o involuntaria, estas características han sido adoptadas tácitamente por el ser humano a lo largo de la historia, tal y como evidencia la siguiente figura. En ella, puede comprobarse la evidente similitud entre la posición adoptada por una bandada de *Leucophaeus Scoresbii* (Gaviota Común) al hacer frente a su principal depredador, el *Aquila Chrysaetos* (Águila Real), con la estrategia militar *en orbe*, de elevada popularidad entre las huestes del Imperio Romano (27 A.C – 476 D.C):

Figura XLVII: Formación de una bandada de *Leucophaeus Scoresbii* frente a la formación militar *en orbe*

Tal y como el lector amante de la naturaleza a buen seguro conocerá, y a pesar de las síncronas y elegantes acciones coordinadas entre individuos de dichos

enjambres, tales conjuntos carecen de un *coordinador* o sistema de control global. No existe un animal que haga las veces de líder en la bandada o banco, sino que el sistema transita en una continua auto-organización, en la que el conocimiento de cada individuo es compartido en pos de averiguar el estado *global* del enjambre.

Durante la historia, los investigadores de los denominados *enjambres sociales* han ejercido un especial énfasis en el rol de los procesos de aprendizaje en dichos modelos, entre los que destacan los trabajos de los ya citados en el presente documento Kennedy y Eberhart. De su obra, se extrae que, en esencia, el comportamiento social ayuda a los individuos a adaptarse a su entorno, a la vez que asegura que obtienen una información superior, mejorada por parte del enjambre e incomparable a la que obtienen mediante medios empíricos, por ende, a través de sus sentidos.

Aunando el espíritu de las características anteriormente citadas, así como el adscrito al algoritmo *Particle Swarm Optimization* y su homólogo *Grammatical Evolution*, nace el paradigma que da nombre a este epígrafe, *Grammatical Swarm*, de la mano de Anthony Barbazon y Michael O'Neill. En un registro más formal, se denomina *Grammatical Swarm* (GS) a la técnica algorítmica que nace como la unión de la técnica *Particle Swarm Optimization* al esquema *Grammatical Evolution* de *mapping* de genotipo a fenotipo, en pos de la generación de *software* en un lenguaje de programación arbitrario.

Concretamente, en la técnica GS, cada partícula o vector de valor real representa las distintas elecciones posibles en cuanto a las reglas de producción de *software* en una gramática BNF. GS se asienta en la representación lineal adoptada en su homólogo GE, que utiliza, como se explica en el epígrafe precedente, las gramáticas como guía de la construcción de *software* correcto desde el punto de vista sintáctico. Numerosos ejemplos de este asentamiento conceptual pueden estudiarse en [Robo, 2018].

Cabe destacar que existe un elenco, en absoluto despreciable, de ventajas en cuanto a la aproximación gramatical a la algorítmica de corte genético, incluyendo la habilidad de codificar numerosas estructuras de datos en las soluciones generadas. En un Algoritmo Genético con una estructura arbórea, un individuo perteneciente al genotipo está restringido a un único tipo, o lo que es lo mismo, cada operador genético que opere en el árbol debe ser capaz de manejar todos los posibles *inputs* de entrada desde los sub-árboles pertinentes. Los símbolos No Terminales en el esquema gramatical que nos ocupa permiten la existencia de múltiples estructuras de datos (u *objetos*, en el paradigma orientado a dichos elementos) en el proceso de concepción del *software*, objetivo final del algoritmo.

Por otro lado, gracias al proceso de *mapping* de genotipo a fenotipo presente en el algoritmo GE y en su homólogo GS, es posible llevar a cabo un proceso de

extrapolación que codifique el dominio del problema concreto en el lenguaje formal a usar en pos de la generación del *software* resolutivo. La adopción de una representación gramática, unida al proceso de *mapping* anteriormente citado, permite modificar fácilmente dicho lenguaje formal para convertirla en un elemento independiente del lenguaje de programación a utilizar. De la misma forma, este factor posibilita variar la heurística de búsqueda que opera en los *bits* que representan el genotipo. Es reseñable que la heurística adoptada por la técnica GE se corresponde, en gran medida, con un GA, mientras que en la técnica GS se utiliza un *array* de valores reales en detrimento de aquella opción. Esta variación desemboca en un nuevo concepto de programación automática basada en el aprendizaje de corte social, denominado *Swarm Programming* (Programación en Enjambre), que presenta la importante característica de estar desprovisto de cualquier tipo de operador genético, los cuales tienden a presentar un coste computacional elevado. Dicho factor motiva, por ende, un incremento en el rendimiento algorítmico.

En cuanto al aspecto numérico se refiere, destaca que GS parte de exactamente la misma base matemática adscrita a PSO, exhaustivamente descrita en el epígrafe 2.1.4 *Particle Swarm Optimization*, añadiendo constantes adicionales en las variables referidas a velocidad y localización, de tal forma que mientras la velocidad máxima oscila en el rango $[-255, 255]$, la posición lo hará en el rango $[0, 255]$. De esta manera, el algoritmo de enjambre pasa a constar de vectores cuya característica principal es un rango real y finito, que permite cuantificar de una forma más fiel la *clasificación* de las partículas en relación a su desempeño en el algoritmo. Por otro lado, y como se ha especificado previamente, GS adopta la técnica de *mapping* característica de GE, en la que los valores reales de los vectores de partícula son redondeados hacia el valor entero más cercano. Gracias a la adopción de los anteriormente citados vectores de longitud preestablecida en el ámbito de la GS, es posible utilizar un número de dimensiones variable durante el proceso de *mapping*, factor que incrementa de forma notable la resiliencia y la flexibilidad del algoritmo. Así, los valores adscritos al vector pueden ser utilizados en más de una ocasión en caso de que se necesite llevar a cabo un proceso de *wrapping*, por lo que es posible que no todas las dimensiones se vean involucradas durante el proceso de *mapping* si un *software* completo (y correcto) es generado de forma previa al alcance del fin del vector. En este último caso, es importante reseñar que los valores *extra* adscritos a la dimensión son ignorados, por lo que se el tiempo de cómputo disminuye de forma directamente proporcionalidad a la cardinalidad del conjunto dimensional.

Por ende, englobando la información ofrecida a lo largo del presente epígrafe sobre el algoritmo GS, es lícito establecer que dicha técnica algorítmica se define a tenor de los siguientes pasos:

1. Aplicación del algoritmo PSO: Se actualiza la posición y velocidad de las partículas, en función de su valor de aptitud, con los *arrays* pertinentes, que serán utilizados en la búsqueda del espacio de búsqueda.
2. Ejecución de la técnica GE: Transformación de los *arrays* que contienen la información de las partículas en *software*.
3. Lectura del lenguaje formal definido en BNF: Gramática que describe las características del lenguaje de programación a utilizar.
4. Generación del programa: El algoritmo crea el *output* pertinente, que se corresponde con el *software* generado.
5. Actualización de los valores de aptitud: Una vez los valores adscritos al *array* son recalculados en base a la función de aptitud, se está en posición de volver al punto 1, dando lugar al comienzo de una nueva iteración.

Tal secuencia algorítmica se refleja fielmente en el siguiente flujograma:

Figura XLVIII: Flujograma adscrito al funcionamiento del Algoritmo *Grammatical Swarm*

Con respecto al apartado más puramente práctico referido al algoritmo GS, destaca, una vez más, el *opus magnum* de Anthony Brabazon y Michael O'Neill; *Grammatical Swarm: The generation of programs by social programming* [Brabazon, 2012]. En dicha obra, se aprecia como los autores abordan numerosos problemas clásicos en base a la técnica GS, abordaje en el que es reseñable la aceptación de los siguientes parámetros:

$$c1 = c2 = 1.0$$

$$wmax = 0.9$$

$$wmin = 0.4$$

$$cmin = 0$$

$$cmax = 255$$

Por ende, adviértase que el mínimo valor que una coordenada podría adoptar (*cmin*) se corresponde con 0 y un valor máximo (*cmax*) de 255.

En el ambicioso estudio anteriormente citado, la batería de problemas se aborda, a su vez, con la técnica GE. De esta forma, se posibilita la comparación empírica entre ambos algoritmos en base a su rendimiento en el desempeño de idénticas tareas. Por otro lado, a fin de lograr la consecución de una justa comparativa, cada algoritmo se restringe en cuanto al número de individuos procesados en base a la siguiente premisa:

- *Grammatical Swarm*: Tamaño de enjambre de 30 procesos, ejecutándose en un millar de iteraciones. (Tamaño genotípico, por ende, de 30.000 individuos).
- *Grammatical Evolution*: Tamaño de población estándar de 500 individuos, que serán alterados en 60 generaciones, alcanzando exactamente el mismo tamaño genotípico.

Por otro lado, los parámetros restantes adscritos al paradigma GE se corresponden con una Selección de Ruleta (*Roulette Selection*), Reemplazo de Estado Activo, Cruzamiento en un Punto con probabilidad de 0.9 y mutación, a nivel de alelo o *bit*, con la ínfima probabilidad de 0.01.

En este momento se está en posesión de abordar el primero de los problemas presentados, que se corresponde con el clásico *Santa Fe Trail* (Camino de Hormigas de Santa Fe). En dicho problema, el objetivo primordial es el de hallar un *software* capaz de controlar un formícido artificial, de tal forma que este pueda encontrar las 89 fuentes de sustento situadas en un camino discontinuo. Por otro lado, existe la restricción de que el número de pasos adscritos al algoritmo es preestablecido, mientras que el escenario se corresponde con una matriz toroidal cuadrada de treinta y dos posiciones. La siguiente figura ilustra un posible escenario adscrito al Camino de Hormigas de Santa Fe [Urbano et al., 2018], donde las posiciones coloreadas en azul simbolizan las distintas ubicaciones donde el alimento reside:

Figura XLIX: Escenario adscrito a la matriz toroidal representativa del problema *Santa Fe Trail*

Adviértase que, en la problemática presentada, la hormiga únicamente está en posesión de girar hacia la izquierda o derecha, avanzar una casilla y *vislumbrar* una posición por delante en la dirección a la que está orientada a fin de determinar si la siguiente casilla del hipotético tablero contiene una fuente de sustento. Todas las acciones anteriormente citadas, a excepción de la última, necesitan de un *paso* algorítmico para ejecutarse. A su vez, es reseñable que la posición inicial del formícido se corresponde con la esquina superior izquierda del escenario.

Retomando la obra anteriormente citada, cabe destacar que el lenguaje formal utilizado figura a continuación:

```

<code> ::= <line> | <code> <line>

<line> ::= <condition> | <op>

<condition> ::= if(food_ahead()) {
                <line>
            }
            else {
                <line>
            }

<op> ::= left(); | right(); | move();

```

Tal y como puede apreciarse en la gramática superior, la estructura de más alto nivel del programa será código (code). A su vez, este elemento podrá estar formado por un objeto línea (line), o bien otro código y una línea adicional. Por otro lado, la línea podrá ser una condición (condition) o bien una operación (op). Mientras que la condición viene descrita por la simple estructura de control if-else que figura en el lenguaje formal superior, la operación será obligatoriamente un movimiento hacia la izquierda (left();), derecha (right();) o hacia delante (move();).

Una vez expuestas las principales características del experimento abordado, se presenta el ilustrativo gráfico adscrito a la ejecución del algoritmo. En dicha representación, el lector puede apreciar gráficamente el mejor valor de aptitud medio, así como la frecuencia acumulativa de éxitos para cada centena de iteraciones. Como puede apreciarse en la figura, la convergencia hacia el mejor valor de aptitud es alcanzada, entendiéndose como valor óptimo de *fitness* el escenario en el que el formícido devora las 89 fuentes de sustento esparcidas a lo largo de la matriz toroidal:

Figura L: Gráfica adscrita a la aptitud media en el problema *Santa Fe Trail* (izq) y frecuencia acumulativa de éxito (der)

Tal y como puede apreciarse en la figura superior, en la casuística concreta que nos ocupa, el algoritmo GE presenta un desempeño ligeramente superior a su competidor GS. Estos resultados serán analizados numéricamente y con más detalle posteriormente en el presente epígrafe.

Por otro lado, a fin de comparar nuevamente el rendimiento entre las dos vertientes algorítmicas anteriormente citadas de forma empírica, se presenta el Problema del Multiplexor. En dicho problema, se pretende, de forma atómica, descubrir una expresión de tipo *Booleana* que se comporte de la misma manera que un multiplexor de tres canales de selección. Se aprovecha la ocasión para recordar al lector que se denomina multiplexor a un circuito combinacional que presenta varias entradas y una única salida de datos. Este componente electrónico está dotado de ciertas entradas de control, que a su vez son capaces de seleccionar una, y sólo una, de las entradas de datos para permitir su transmisión desde la entrada seleccionada hacia dicha salida.

Como dato anecdótico, es reseñable que en el campo de la electrónica el multiplexor se utiliza como dispositivo susceptible de recibir varias entradas y transmitir las por un medio de transmisión compartido. Para ello, este componente lleva a cabo una escisión del medio de transmisión en múltiples canales, de tal forma que varios nodos puedan comunicarse al mismo tiempo.

En cuanto a los aspectos referidos al funcionamiento de un multiplexor se refiere, destaca que dichos circuitos combinacionales poseen 2^n líneas de entrada de datos, una línea de salida y n entradas de selección. Estos últimos *inputs* indican cuál de estas líneas de entrada de datos es la que proporciona el valor a la línea de salida, típicamente denominada *Z* ó *out*. Cada combinación de las entradas de selección se asocia a una entrada de datos, mientras que la salida final del multiplexor se corresponderá al valor la entrada seleccionada. A continuación se presenta una figura ilustrativa del aspecto, a nivel lógico, de un multiplexor de una única entrada de selección (*se1*). Adviértase que dicho circuito combinacional puede sustituirse por un conmutador controlado por una señal homóloga *se1*:

Figura LI: Esquema lógico de un multiplexor 2 a 1 con una entrada de selección y del equivalente conmutador controlado

Retomando el Problema del Multiplexor anteriormente citado, y a fin de introducir una enjundia superior en el mismo para que arroje unos resultados más relevantes en cuanto al desempeño de los algoritmos objetos de estudio, se tomará un multiplexor de tres entradas de selección (ocho canales, por ende), como objetivo. Por tal motivo, existen ocho valores de aptitud o *fitness* para el escenario concreto, representando la totalidad de pares *input-output* existentes. A tenor de este razonamiento, la aptitud será correspondiente al número de casos de entrada para los que la expresión *evolucionada* devuelve el *output* pertinente. La gramática utilizada en el problema descrito figura a continuación:

```

<mult> ::= guess = <bexpr>;

<bexpr> ::= (<bexpr> <bilop> <bexpr>)
          | <ulop> (<bexpr>)
          | <input>

<bilop> ::= and | or

<ulop> ::= not

<input> ::= input0 | input1 | input2

```

El lenguaje formal anteriormente descrito se corresponde exactamente con el funcionamiento de un multiplexor de tres entradas de selección, por lo que, persiguiendo la finalidad de evitar redundancias, se obviará su explicación exhaustiva. Tras presentar dicho lenguaje se está en condiciones de mostrar la gráfica asociada al comportamiento del algoritmo GS y su competidor GE en cuanto a la media alcanzada del valor de aptitud y la frecuencia acumulativa de éxitos:

Figura LII: Gráfica adscrita a la aptitud media en el Problema del Multiplexor (izq) y frecuencia acumulativa de éxito (der) [Urbano et al., 2018]

Tal y como los resultados arrojan, el estudio de la gráfica indica que, al contrario que en el problema de *Santa Fe Trail*, el algoritmo GS presenta un rendimiento superior a su competidor, GE.

Finalmente, se presenta la tabla que aúna los resultados anteriormente mostrados a modo de gráfica para ambos problemas (los mejores valores aparecen resaltados a fin de facilitar la visualización de los mismos). Cabe destacar que el experimento en cuestión se ha desarrollado utilizando un conjunto de 100 ejecuciones sobre el *benchmark* matemático que aparece en [Urbano et al., 2018], y ha sido ejecutado en un *MacBook Pro* con procesador *Intel Core i7* de cuatro núcleos, a 2,9 GHz con *Turbo Boost* de hasta 3,9 GHz, 8 MB de caché de nivel 3 compartida y un *SSD PCIe* integrado de 512 GB:

	Mejor valor de aptitud medio	Desviación Estándar	Mediana	Ejecuciones exitosas
<u><i>Santa Fe Ant Trail</i></u>				
GS	75.24	16.64	88.00	43
EG	<u>80.18</u>	<u>13.79</u>	<u>89.00</u>	<u>58</u>
<u><i>Multiplexor</i></u>				
GS	<u>0.97</u>	<u>0.05</u>	<u>1.00</u>	<u>79</u>
EG	0.95	0.06	1.00	56

Figura LIII: Resultados arrojados tras la comparativa empírica entre el algoritmo GS y su competidor GE

Como puede apreciarse tras una somera visión panorámica, pues la información adscrita al algoritmo que mejor rendimiento presenta en cada caso aparece resaltada y subrayada, existe un empate en los resultados. El algoritmo GE disfruta de un rendimiento superior en el problema *Santa Fe Trail*, mientras que no ocurre lo mismo en el Problema del Multiplexor, donde su homólogo GS lidera la tabla en cuanto al rendimiento algorítmico.

Se recuerda que, persiguiendo la finalidad de obtener una comparativa ecuánime, se han utilizado conjuntos genotípicos de la misma cardinalidad para cada escenario, por lo que el procesamiento *software* se ha llevado a cabo sobre un número idéntico de individuos.

Por tanto, se concluye que la principal inferencia extraída a tenor de la luz que arrojan los resultados es que el algoritmo GS se eleva como una más que digna

alternativa a GE, pues es capaz de generar soluciones satisfactorias de igual manera e incluso presentando un mayor rendimiento en ciertos casos. Por ende, en la humilde opinión del doctorando, es recomendable realizar una prueba de concepto similar a la anteriormente citada en función del problema, ya que no existe, hasta la fecha, ningún otro criterio de selección algorítmica al respecto. La comparativa en cuanto al rendimiento en el experimento inicial, indicará qué algoritmo debe ser aplicado en pos de velar por un desempeño computacional óptimo.

2.1.7 Memetic Computing

Desde un punto de vista etimológico, un *meme* [Oxford, 2018] puede definirse como un elemento de una cultura o sistema de comportamiento que fluye de un individuo a otro por imitación, así como por otros medios de transmisión no genéticos.

En la misma línea, los sufijos tico/tica (del latín *-ticus* y este del griego antiguo - τικός, *tikós*), pueden traducirse como “relativo a”. Aunando dicha definición con la anteriormente presentada, se está en posición de inferir que la memética es el concepto que aglutina todo lo relacionado con el trasiego de información unidireccional, de progenitor a descendencia, entre individuos.

En otras palabras, Richard Dawkins define la memética como una “hipótesis de contenido mental basado en una analogía de la evolución darwiniana” [Dawkins, 2000]. La memética puede concebirse como una aproximación a los modelos evolutivos de transferencia de información cultural, entendiendo el meme como la unidad de cultura que se hospeda en la conciencia colectiva.

Tal y como ocurre con numerosos elementos en el mundo de la informática, el anterior concepto encuentra su alter ego computacional en el paradigma de la Computación Memética.

En el ámbito de dichos paradigmas, los Algoritmos Meméticos (AM en lo sucesivo) [Moscato, 1989] representan una de las más novedosas áreas de investigación de la Computación Evolutiva. Un AM puede definirse como cualquier esquema evolutivo basado en población que separa directamente el aprendizaje de cada individuo en la búsqueda de la solución. En otras palabras, puede inferirse que un AM no trata la población como un colectivo, como PSO, sino que la divide en elementos atómicos e independientes.

Es reseñable que los AM son con frecuencia referenciados mediante varios sinónimos [Chen et al., 2011], entre los que destacan Algoritmos Evolucionarios Baldwinianos (EA), Algoritmos Evolucionarios Lamarckianos, Algoritmos Culturales o Búsqueda Local Genética.

En cuanto a su definición se refiere, el término AM viene inspirado por los principios de evolución Darwiniana, así como por la etimología de meme establecida por Dawkins. Aunando ambas, Moscato define los AM como una estructura cercana a los Algoritmos Genéticos basados en el análisis de la población que recibe como parámetro, complementados con un proceso de aprendizaje capaz de llevar a cabo un refinamiento local.

Adviértase aquí el gran paralelismo con la evolución Darwiniana por una parte, así como con las heurísticas propias de la naturaleza de los memes. Esta sinergia da lugar a una metodología que obtiene un balance entre la generalidad y la

especificidad de los problemas que aborda, desembocando así en un carácter de Fase Dual Evolutiva de los AM.

Puede, por tanto, inferirse que todos aquellos algoritmos pertenecientes a este espectro son englobados por el paradigma de la Computación Memética (*Memetic Computation*; CM). Dicho paradigma tiene por objetivo el estudio de las áreas de los algoritmos evolutivos que necesitan de otras técnicas de refinamiento para la solución de problemas de optimización, extendiendo la noción de los memes al de entidades conceptuales sobre procedimientos o representaciones cognitivas [Merz, 2002].

Por otro lado, en cuanto a la taxonomía de los AM se refiere, destacan tres grandes grupos o generaciones [Moscato, 1989] en los que se dividen:

1. AM híbridos: La primera generación de AM se corresponde a un conjunto híbrido, fruto de una sinergia entre la búsqueda global basada en población y la evolución cultural mediante el uso de memes. A pesar de que los AM híbridos engloban características fundamentales de la evolución cultural en el ciclo de búsqueda, es reseñable que no son definibles como un sistema evolutivo puro en base al Darwinismo Universal. Este factor es debido a que las principales características de la transmisión genética y herencia no están presentes en esta generación de algoritmos. A continuación se presenta el pseudocódigo en base al cual los AM operan [Moscato, 1989]:

```
createInitialPopulation();
while (!stoppingConditions)
{
    evaluateAllIndividuals();
    evolveNewPopulation(stochasticSearchOperators);
    individuals = selectIndividualsToImprove();
    for(allIndividuals) {
        performLearningProcess(meme, isLamarckianLearning,
isBaldwinianLearning);
    }
}
```

2. AM Meta-Lamarckianos [Ong et al., 2004]: La principal diferencia en esta generación, que recibe su nombre del biólogo Jean Baptiste Lamarck, radica en la presencia de los principios de transmisión memética y selección en su diseño. En este subgrupo algorítmico, el material memético es codificado directamente como una sección del genotipo. De esta forma, el meme decodificado de cada cromosoma será usado como argumento en el proceso de refinamiento que se llevará a cabo sobre la población. El material memético será distribuido a toda la descendencia cromosómica partiendo de un esquema de herencia simple, donde cada

cromosoma descendiente presentará alelos del progenitor. Es digno de mención que, con respecto a los AM Meta-Lamarckianos, el conjunto de cromosomas candidatos competirán entre sí, basado en sus hitos pasados y en un sistema de puntuaciones, persiguiendo la finalidad de hacer persistir los memes que poseen hacia las próximas generaciones. En este proceso, se verifica que aquellos memes que presenten una puntuación mayor obtendrán una posibilidad mayor de ser traspasados a la siguiente generación.

3. AM Coevolutivos [Smith, 2007]: Pertenecen a este subconjunto generacional todos aquellos AM en los que se verifique que las tres condiciones de un sistema evolutivo básico se cumplen. Además de esta condición triple, debe cumplirse que, en contraste con la segunda generación de AM, los memes a utilizar en las iteraciones algorítmicas sean conocidos de forma previa a la ejecución de las mismas. Los AM Coevolutivos utilizan, por otro lado, un sistema de búsqueda local basado en reglas para sustituir las soluciones candidatas del sistema evolutivo, por consiguiente analizando los patrones que acontecen en el espacio del problema a resolver.

A continuación se presenta un diagrama de flujo que representa el funcionamiento de los AM:

Figura LIV: Diagrama de Flujo de un Algoritmo Memético

Una vez definidos los AM, es recomendable mencionar ciertas asunciones a tener en cuenta a la hora de utilizarlos. La frecuencia, así como la intensidad con la que cada aprendizaje individual se lleva a cabo, es responsable de la definición directa de la carga computacional del AM, por lo que es un aspecto a tener crucialmente en cuenta, sobretodo si el AM debe ser ejecutado bajo ciertos umbrales de rendimiento. Claramente, un aprendizaje cromosómico más intenso en cómputo otorga una posibilidad mayor de convergencia algorítmica hacia un óptimo local, pero limita la posibilidad de evolución sin incurrir en un uso exacerbado de recursos. Por esta razón, los parámetros anteriormente citados han de tenerse muy en cuenta para alcanzar un equilibrio entre el presupuesto computacional y la necesidad de análisis de la mayor cantidad posible de espacio de búsqueda. Cuando únicamente un subconjunto genotípico goza del privilegio de llevar a cabo un aprendizaje, el problema de qué individuos deben disfrutar de dicha ventaja debe considerarse para velar por un carácter óptimo del AM. Por último, el meme utilizado por cada individuo debe velar por cómo será transmitido a sucesivas generaciones, siendo necesaria la definición de dicho trasiego en el

proceso de definición del AM. Además de las ya citadas características, conviene no perder de vista aspectos clave como:

- Frecuencia de aprendizaje cromosómico: La cadencia con la que los cromosomas que componen la población llevan a cabo el proceso de asimilación de los memes supone uno de los principales factores a tener en cuenta en el diseño de un AM. En [Hart, 1994], puede estudiarse cómo el efecto de la frecuencia de aprendizaje individual para el AM fue ejecutado en varias etapas del algoritmo. Los resultados arrojan que, en caso de que la complejidad computacional asociada a los memes sea baja, es recomendable llevar a cabo el aprendizaje cromosómico de forma individual.
- Soluciones susceptibles de utilizar aprendizaje individual: Las funciones fitness, tratadas pormenorizadamente en la presente Tesis Doctoral en lo respectivo a los GA, constituyen un aliado a la hora de valorar si el aprendizaje individual es viable en el ámbito de un AM.
- Frecuencia del aprendizaje individual: Persiguiendo la finalidad de acotar el coste computacional que supone el proceso de aprendizaje, es recomendable asignar una cota máxima asociada a cada cromosoma. De esta forma, en escenarios con memes de complejidad variable, cada individuo tendrá asociado un valor máximo de tiempo de cómputo, que evita cuellos de botella algorítmicos en la ejecución del AM.
- Criterio de selección del proceso de aprendizaje individual: En el contexto de la optimización continua, el aprendizaje individual se presenta en la forma de procesos heurísticos locales, o bien de métodos probabilísticos exactos [Schweffel, 1995]. A modo de ejemplo ilustrativo en cuanto a los múltiples tipos de aprendizaje individual existentes, destaca el método Quasi-Newtoniano, el del Punto Interior, Gradiente Conjugada, Búsqueda Lineal y Simplex, por citar algunos. Es reseñable que la mayoría de los métodos de aprendizaje común son de tipo determinista e idempotente, por lo que el aprendizaje del mismo meme por parte de un cromosoma, producirá siempre el mismo resultado.

Una vez definido el concepto de AM, así como sus principales características y variaciones, es digno de mención el elenco de escenarios donde es aplicable. En dicho sentido, destaca que los AM han sido aplicados con éxito a un amplio espectro de problemáticas asociadas al mundo real. A pesar de que un gran número de las investigaciones referidas a los AM se centran en el carácter más puramente conceptual de esta rama algorítmica de la Computación Natural, destacan esquemas híbridos entre los AM y los GA. De esta forma, el conjunto genotípico del GA se ve sometido a un proceso de aprendizaje en el que cada cromosoma aprende un meme, que recibe como parámetro. Esto permite ajustar

la función fitness, obteniendo resultados óptimos o, al menos, más cercanos a la solución real.

Por otro lado, es digna de mención la extensa aplicación de los AM a los problemas NP clásicos, como la Partición de Grafos, el Problema del Viajante, la Asignación Cuadrática y un largo etcétera.

Además, los AM se han aplicado con éxito a esquemas de gestión de nurse rostering en el contexto hospitalario [Ozcan, 2007], entrenamiento de redes neuronales artificiales [Ichimura et al., 1998], gestión del movimiento de corte robótico [Ridao, 1998], diseño VLSI [Areibi et al., 2004] y la selección multi-objetivo [Karkavitsas et al., 2011].

Finalmente, y una vez definido el concepto de AM, así como sus principales características, destaca una biblioteca software que otorga al desarrollador un marco completo para la implementación de AM; *jMeme* [Acampora et al., 2016].

En dicha publicación, cuna de *jMeme*, se presenta, en primer lugar, a los AM como un conjunto de metaheurísticas puro que combina esquemas de búsqueda global con técnicas homólogas a nivel local. Esta aproximación híbrida han hecho emerger a los AM como poderosas herramientas para resolver problemas de optimización complejos de mejor forma que sus homólogos no híbridos. Sin embargo, y a pesar de dicho éxito, el diseño de los AM presenta un importante y elevado número de problemas de diseño que deben resolverse en pos de alcanzar una solución precisa del problema de optimización a resolver. A excepción de *jMeme*, no existe ninguna herramienta software que permita diseñar AM, y, por consiguiente, todas las decisiones que presenta la hibridización quedan a discreción del investigador, que como ser humano puede equivocarse. El objetivo de *jMeme* es, precisamente, salvar dicho vacío mediante la toma de decisiones automatizada. Tal y como demuestran los experimentos llevados a cabo en el artículo de Acampora anteriormente citado, el diseño de AM por parte de *jMeme* presenta un rendimiento mayor que aquellos AM convencionalmente diseñados.

Dicha biblioteca permite llevar a cabo cualquier tipo de diseño enmarcado en el ámbito de los AM, así como configuraciones de tipo híbrido para problemas de ámbito múltiple.

Basado en el lenguaje de programación Java, *jMeme* otorga una serie de mecanismos que apoyan la integración entre búsquedas locales y globales de una forma directa y simple. Es reseñable que, de forma nativa, *jMeme* incluye métodos de búsqueda global estudiados en la presente Tesis Doctoral, tales como GA, ED, PSO y ACO.

Desde un punto de vista software, *jMeme* puede definirse como un conjunto de clases abstractas e interfaces que modelan totalmente los conceptos generales

circunscritos al ámbito de los AM, así como métodos locales que otorgan un amplio grado de extensibilidad al software desarrollado mediante *jMeme*.

A modo de ejemplo de uso de dicha biblioteca, se presenta la resolución del *Napkin Ring Problem* o *Sphere Problem*. En dicho problema, se pretende encontrar el volumen de una banda de una altura específica alrededor de una esfera:

Figura LV: Áreas representando el volumen adscrito al *Napkin Ring Problem*

A modo de ejemplo, se destaca que, gracias a *jMeme*, el volumen de altura aleatoria asociado a un *Sphere Problem*, también aleatoriamente generado, puede resolverse mediante el siguiente código fuente, desarrollado en el ámbito de la Tesis Doctoral que nos ocupa y disponible en <https://github.com/ClemencioMoralesLucas/PhD>:

```

public class TestCompetentStudyXSphere {

    public static void main(String[] args) {
        SphereProblem pSphere = new SphereProblem( name: "Sphere", num: 2);
        SphereFitness fSphere = new SphereFitness();

        FitnessFeatures fitnessSphere = new FitnessFeatures(fSphere, FitnessFeatures.MIN);
        Problem[] pp = new Problem[1];
        pp[0] = pSphere;

        AlgorithmFinalConditions finC = new AlgorithmFinalConditions(FinalConditions.FINAL_CONDITION_FITNESSEVALUATIONS, value: 1500);
        AlgorithmComponent[] gcc = new AlgorithmComponent[2];
        AlgorithmComponent[] lcc = new AlgorithmComponent[2];

        GParameters ga = new GParameters( mut: 50, new CrossoverParameters( rate: 0.9, CrossoverParameters.BLXalpha3));
        gcc[0] = new AlgorithmComponent( id: "ga", GA_Optimizer.class.getName(), ga);

        PSOParameters pso = new PSOParameters();
        gcc[1] = new AlgorithmComponent( id: "pso", PSO_Optimizer.class.getName(), pso);

        StochasticHillClimbingParameters st = new StochasticHillClimbingParameters();
        lcc[0] = new AlgorithmComponent( id: "st", StochasticHillClimbing.class.getName(), st);

        SimulatedAnnealingParameters sa = new SimulatedAnnealingParameters();
        lcc[1] = new AlgorithmComponent( id: "sa", SimulatedAnnealing.class.getName(), sa);
        CompetentStudy proc = new CompetentStudy(pp, fitnessSphere, popSize: 20, finC, gcc, lcc, new int[]{10, 30, 50},
            new double[]{0.05, 0.5, 1.0}, new String[]{LocalOptimizerExecutor.BEST_SELECTION_SCHEME,
                LocalOptimizerExecutor.RANDOM_SELECTION_SCHEME, LocalOptimizerExecutor.STRATIFIED_SELECTION_SCHEME});
        proc.execute();
    }
}

```

Figura LVI: Método principal para la resolución de *Napkin Ring Problem* con *jMeme*

Al ejecutar dicho algoritmo, podemos hallar los resultados de la simulación en la siguiente captura de traza:

```

Iterations 50 Evaluations 50
best fitness value after local search: 3.897816830315198E-9
number of evaluations: 1496
iteration ended
Iteration number 18
best fitness value after evolution: 3.897816830315198E-9
best fitness value after local search: 3.897816830315198E-9
number of evaluations: 1500
iteration ended
time of executions: 4
evaluations: 1500
best      Size:2,      Fitness      value:3.897816830315198E-9,      Alleles:[DoubleGene(-
5.12,5.12)=3.136813215472844E-5,      DoubleGene(-5.12,5.12)=5.398015482970283E-5],      Application
data:null
Runs 10 executed
Test terminated on configuration Stratified

Global Component: ga
Global Component features:
Crossover type: BLXalpha3
Crossover rate: 0.9
Mutation type: Uniform
Mutation rate: 0.02
Selector type: Best Chromosomes
Selector rate: 0.9

Local Component: st

```


Local Component features:

Max distance: 1

Local Intensity: 50

Local Frequency: 0.05

Selection scheme: Random

Common features:

Population size: 20

Termination:

Number of evaluations 1500

Configuration Serialized

Process finished with exit code 0

Tal y como ha podido apreciarse a lo largo del presente epígrafe, la AM se erige como uno de los principales pilares modernos de la Computación Natural, uniéndose al grupo de algoritmos y paradigmas presentados en la Tesis Doctoral que nos ocupa.

Por tanto, a modo de conclusión en lo referido a los AM, se destaca un amplio nivel de personalización como una de sus principales virtudes. No sólo estos algoritmos son susceptibles de ser aplicados a un gran número de problemáticas de diversa índole, sino que además pueden unirse con otros paradigmas adscritos al ámbito de la Computación Natural. De esta forma, los AM permiten lograr poderosas sinergias al complementarse con Algoritmos Genéticos o Optimización por Enjambre de Partículas, estudiados pormenorizadamente en la presente Tesis Doctoral. Esta característica de los AM refuerza un hecho constante en la Computación Natural, y es que la mayoría de sus algoritmos pueden ser unidos para trabajar en pos de la consecución de un objetivo común. Dicha característica, unido al amplio elenco de problemas resolubles mediante AM, los erige como una poderosa herramienta algorítmica, especialmente en el ámbito de los problemas de optimización.

2.2 Estado del arte del problema que aborda la Tesis Doctoral

Dado que en el capítulo 2. *Estado del Arte* se han descrito exhaustivamente las condiciones teóricas actuales de *Fireworks Algorithm* (FA) y *Grammatical Evolution* (GE), entre otros, el presente subepígrafe se centrará en el Estado del Arte orientado a la investigación desarrollada en la presente Tesis Doctoral. Por consiguiente, se identificarán los distintos retos que FA y GE enfrentan, futuras investigaciones de interés e ideas a desarrollar en sus respectivos ámbitos.

2.2.1 Grammatical Evolution

A tenor del carácter novedoso de GE, existen numerosos retos abiertos actualmente, tal y como demuestra [O'Neil, 2010]. Los más importantes, varios de ellos relacionados con *Genetic Algorithms* (GA), se detallan seguidamente:

- Inexistencia de estudios actuales que apliquen *Ant Colony Optimization* (ACO) en el ámbito de GE: A pesar de la existencia de investigaciones que aplican otras técnicas de la Computación Natural en GE, como [Koza, 1989], ninguna toma ventaja de los posibles beneficios de ACO. (Este factor es precisamente conseguido por el Algoritmo ACORD [de Mingo et al, 2019] que propone la presente Tesis Doctoral.)
- Escalabilidad de GE: Actualmente, no existe ninguna directiva estándar a respetar para conseguir escalabilidad en GE. Debe recordarse el deseable carácter de esta propiedad, que denota la habilidad del algoritmo para mantener su fluidez cuando la carga de trabajo aumenta. La investigación fructífera en esta área permitiría a GE poder lidiar con cargas de trabajo colosales sin que su rendimiento se viera comprometido.
- Identificación de representaciones apropiadas, especialmente cuando se intentan unir GE y los denominados GA: Hasta el momento, la detección de criterios de calidad ideales, en términos de aptitud, que beneficien el proceso de búsqueda de GE al mismo tiempo se resiste a ser definida. A pesar de que esta identificación se antoja difícil, un mejor entendimiento de la relación entre representación y búsqueda en GE podría posibilitar el proceso [Fogel, 1998].
- Rendimiento de problemas en entornos dinámicos: La diversidad de los problemas dinámicos a resolver mediante un enfoque híbrido GE-GA, dado el número de cambios en el problema y su carácter, dificulta la estimación

en términos de rendimiento. Este hecho se traduce en la posibilidad de que el algoritmo presente un rendimiento paupérrimo en ciertos escenarios.

- Dificultad de establecer el nivel de detalle, en términos generales, para adoptar fenómenos de corte biológico, especialmente en GE: El grado de abstracción y simplificación a la hora de simular ciertos comportamientos del mundo biológico es, actualmente, difícil de conocer de antemano. La creación de bases para estándares futuros en este aspecto podría suponer un punto de inflexión muy interesante, discerniendo causas y efectos para distintos niveles de detalle.
- Sistemas basados en GE-GA evolutivos con independencia (*Open-Ended Evolution*): Un auténtico reto en el ámbito de ambos algoritmos se corresponde con la posibilidad de crear sistemas evolutivos en sí mismos, sin necesidad de ninguna modificación humana para que esto ocurra. Por citar un ejemplo, un sistema GE-GA que dé cuerpo al núcleo de un sistema de *trading*, y que sea capaz, en sí mismo, de aprender y adaptarse a lo largo del tiempo incluso a problemas que difieren del original. Un interesante estudio basado en esta línea puede consultarse en [Moore, 2008].
- En ciertos problemas híbridos GE-GA, especialmente en aquellos casos donde existen varias variables de interés, la generalización no es trivial. Este factor desemboca en que el proceso de decisión sobre qué debe exactamente medir la función de aptitud debe ser experimental. En otras palabras, se antoja necesario realizar pruebas con distintas funciones de aptitud antes de elegir la definitiva. Cualquier investigación en la línea de generalizar este proceso, convirtiéndolo en teórico más que experimental, agilizaría notablemente el proceso.

Presentados algunos de los retos más importantes a los que se enfrenta GE, destaca a su vez que, a pesar de la juventud de la técnica, existen ya varias versiones mejoradas y variantes, especialmente en el ámbito de *Particle Swarm Optimization* (PSO) [Miranda, 2016]. En dicho estudio, PSO se utiliza para llevar a cabo el proceso de búsqueda, por lo que esta mezcla GE-PSO recibe el término *Grammatical Swarm*, como se ha especificado en su epígrafe dedicado en la presente Tesis Doctoral. En términos de rendimiento, es destacable que Grammatical Swarm es equiparable al híbrido entre *Genetic Algorithms* y GE.

Por otro lado en cuanto a GE, se destaca que, dado su Estado del Arte actual, se espera un continuado crecimiento de la técnica, principalmente gracias a las deseables características que otorgan las gramáticas, siendo la generalidad la principal: En el caso Algoritmos Genéticos, disponer de más información sobre el problema presenta, en numerosas ocasiones, un problema más que una solución,

y es que no existe una forma directa o trivial de incorporar esta información al algoritmo sin modificar su código fuente. No obstante, esta información sí puede utilizarse para modificar una gramática de entrada, elevando la potencia de Grammatical Evolution sin sacrificar su código. En palabras de [O'Neill, 2001]: “*The more one knows about a problem, the more one can enrich the grammar with all manner of insider information about the problem.*” (Conforme más se sabe del problema, más se puede enriquecer la gramática con información privilegiada sobre el problema.). Además, la salida del programa puede llevarse a cabo en cualquier formato que pueda codificarse en una gramática, como demuestra, a modo de ejemplo [Ryan et al., 1998]. Otra poderosa ventaja de las gramáticas se corresponde con el concepto de *validez* de las mismas, puesto que una gramática bien formada siempre producirá como salida una instancia válida. Por si fuera poco, existe también la posibilidad de codificar información semántica en la propia gramática, a fin de intentar sesgar (de manera positiva para el desempeño del programa) el espacio de búsqueda.

Atendiendo al Estado del Arte actual, la investigación futura en GE y posiblemente otros métodos basados en el uso de gramáticas, también deberían poseer especial interés en dos áreas complementarias: las modificaciones en el proceso de *mapping* y la adaptación de métodos existentes a las necesidades de GE. (Tal y como se lleva a cabo precisamente en esta Tesis Doctoral con *Ant Colony Optimization* aplicado a GE.) Es importante destacar, una vez más, que la elección de la gramática delimita el espacio de búsqueda. De hecho, gramáticas equivalentes se diferenciarán únicamente en su preferencia por unas u otras estructuras, típicamente no terminales. Esta característica se subraya en [O'Neill, 2001], donde se sugiere una co-evolución dinámica de las gramáticas propuestas como línea de investigación futura al estudio.

De forma paralela, otro interesante reto al que se enfrentan las investigaciones actuales alrededor de GE se corresponde con un curioso escenario que acontece al aplicar los operadores genéticos de los algoritmos homónimos, y es que en ocasiones se produce el denominado “*efecto de onda*” (*ripple effect*). En tal sentido, la experiencia sugiere que dicho fenómeno promueve un intercambio útil de sub-secuencias de derivación durante cruzamientos genéticos [Dempsey, 2009], contribuyendo en igual medida a la deseable diversidad fenotípica en este tipo de algoritmos.

Finalmente, conviene recordar otra desventaja importante de GE, cuya resolución sería especialmente beneficiosa para el Estado del Arte actual de la técnica, y es que, en ciertas ocasiones, la hibridación de un *Genetic Algorithm* con *Grammatical Evolution* es ciertamente problemática. Esta desventaja proviene, principalmente, del amplio coste computacional que conlleva aplicar un GA a una gramática que genere un árbol de altura suficiente para ciertos *inputs*. Interpretando como individuos pertenecientes al genotipo del GA las numerosas y

altamente probables variaciones que pueden acontecer al evaluar la gramática en un caso concreto, se recuerda que GA es intensivo en cálculo, ya que para hallar el fenotipo final llevará a cabo un alto número de variaciones, permutaciones, recombinaciones, cruzamientos y otros operadores genéticos sobre el genotipo. De esta manera, la complejidad (y tiempo de ejecución) al combinar GA y GE podría crecer de manera exponencial siempre y cuando no se controle de alguna manera. (Precisamente este control se efectuará gracias al valor de *threshold* en el caso de ACORD en esta Tesis Doctoral.)

2.2.2 Classic Fireworks Algorithm

El algoritmo FA, como se ha especificado pormenorizadamente en este mismo documento, es una técnica prometedora que pertenece al Paradigma de la Computación Natural. Especialmente durante la última década, numerosos algoritmos de optimización y búsqueda han visto la luz, pero, desafortunadamente y a tenor del carácter novedoso de FA, no han sido muchos los artículos científicos y publicaciones centrados en su optimización.

La mayoría de estos trabajos se centran, primordialmente, en la explicación del algoritmo y en sus aplicaciones, por lo que, más que la búsqueda de híbridos entre FA y otras entidades adscritas al Paradigma de la Computación Natural, es interesante centrarse en la optimización pura del algoritmo.

Existen pocos estudios comprensivos comparando los algoritmos de búsqueda y optimización. Es reseñable destacar, a título ilustrativo, “*Comparison of local search algorithms with population-based algorithms in hybrid flow shop scheduling problems with realistic characteristics*” [Bozorgirad et al., 2016] como uno de los mejores estudios conocidos. De nuevo, estudios como este se centran en la aplicación algorítmica, pero no en su mejora.

A pesar de que FA apenas cuenta con dos años de vida en el momento en el que se escribe esta Tesis Doctoral, existen numerosos artículos, como se especificó anteriormente, que se centran en el concepto algorítmico de FA *per se*. No obstante, lo mismo no puede decirse de la mejora algorítmica de la propia entidad, que prácticamente no ha variado. Tal razón puede verse como la principal motivación y justificación para enfocar los esfuerzos investigadores en tal sentido, como se describirá en los epígrafes sucesivos. A modo de avance, se destaca que el Estado del Arte actual de FA parece apuntar a la necesidad de mejora de cuatro grandes áreas del algoritmo, a saber:

1. Explosión de las *sparks*: Recordando que FA define un proceso de búsqueda en un espacio colosal, principalmente tras la explosión de los propios *fireworks*, proceso en el que se crean las *sparks*, el tipo de

explosión es un factor clave en el desempeño del algoritmo. Como se detallará en las secciones dedicadas a *Improved Fireworks Algorithm*, la aplicación de los valores extraídos de una Distribución Gaussiana para variar la dirección de las *sparks* será un punto de inflexión en el rendimiento del algoritmo, puesto que este mecanismo permite *intuir* a IFA dónde podrían, desde un punto de vista estadístico, ubicarse los puntos de interés en el área de búsqueda. La investigación de alternativas de índole similar para la explosión podrían seguir abriendo originales y prometedoras líneas de investigación en FA.

2. Aplicación de técnicas algorítmicas de apoyo: Dada la complejidad de FA, existen numerosos métodos de optimización y mejora *software* susceptibles de ser aplicados; más en concreto, en el caso de IFA, este añadido se lleva a cabo con Divide y Vencerás y Desenroscado de Bucles. Además de esta dupla de efectivas técnicas, la legibilidad y eficiencia del algoritmo podría ser mejorada gracias a la aplicación de los útiles Patrones de Diseño Software [Gamma, 2011], entre los que se destaca *Flyweight Design Pattern* especialmente para la creación de *sparks* gastando pocos recursos computacionales. (Se entiende como *flyweight* o “peso ligero” un objeto que minimiza el uso de memoria mediante la compartición de la mayor cantidad de datos posible con objetos similares, de manera que provee una forma de utilizar grandes cantidades de objetos con una representación simple sin que impacte negativamente en la memoria del sistema.)
3. Decremento de la Complejidad Algorítmica (*Big-O Notation*): La complejidad de FA, que como se describirá en epígrafes sucesivos es cuadrática, no es en absoluto desdeñable. La simplificación del algoritmo, en términos de implementación, puede agilizar el proceso de ejecución, siendo una característica especialmente deseable en la persecución de un mejor rendimiento general para FA. Por consiguiente, los esfuerzos investigadores en esta área son especialmente deseables dado el estado del arte de FA.
4. Reducción de la complejidad espacial: En un algoritmo de implementación tan extensa como FA, existen numerosas variables que pueden ser reutilizadas en ciertas secciones, así como (especialmente) instrucciones que podrían ser redundantes, que podrían optimizarse o bien, directamente, eliminarse en ciertos casos concretos, llevando a cabo las operaciones pertinentes en partes anteriores de la ejecución, donde son más aconsejables en términos de eficiencia. Dada la juventud de FA, el análisis de su complejidad especial consta, por tanto, de un especial interés.

3. ACORD ALGORITHM & IMPROVED FIREWORKS ALGORITHM. METODOLOGÍA

Tal y como se especificó en el epígrafe anterior, se presentará en este epígrafe la doble piedra angular de la presente Tesis Doctoral: El Algoritmo ACORD (**Ant Colony Optimization Rule Derivation Algorithm**) [de Mingo et al, 2019], de índole más teórica y una variación de FA, denominada IFA (*Improved Fireworks Algorithm*; Algoritmo de los Fuegos Artificiales Mejorado), que será empíricamente enfrentado a FA para analizar los resultados que arroje el experimento, de carácter más práctico, ya que la mayoría de trabajos de investigación relacionados con FA están mayormente basados en los aspectos más puramente teóricos de dicho algoritmo.

3.1 Ant Colony Optimization Rule Derivation Algorithm (ACORD)

Es lícito inferir que el paradigma de la computación bioinspirado es más que extenso [Campelo, 2018] [de Castro, 2007]. A pesar de que la ciencia y la ingeniería han evolucionado durante numerosos cientos de años con multitud de herramientas y métodos disponibles para la resolución de problemas, existe aún un diverso rango de problemas a resolver, fenómenos a sintetizar y preguntas a responder. En términos generales, los enfoques que brinda la Computación Natural deben considerarse cuando el problema es complejo y no lineal, e involucra un gran número de variables, soluciones potenciales, consta de múltiples objetivos o el problema a solucionar no puede ser modelado en base a aproximaciones clásicas como reconocimiento complejo de patrones y tareas de clasificación. También puede utilizarse la Computación Natural en aquellas casuísticas donde encontrar una solución óptima mediante enfoques tradicionales no es posible, es difícil de obtener o no puede garantizarse, pero existe una medida cualitativa que permite la comparación de varias soluciones. Así, el problema diverge hacia una diversidad de soluciones, que es especialmente deseable en tales escenarios.

Los algoritmos de optimización inspirados en la naturaleza [Yang, 2016] se categorizan recursivamente en los algoritmos basados en *Swarm Intelligence* y *Evolutionary Intelligence*. Mientras estos últimos se centran en el comportamiento evolutivo de los sistemas naturales, como son los Algoritmos Genéticos, aquellos optimizan ciertos problemas mimetizando el comportamiento colectivo de un *enjambre* natural, como ocurre con *Particle Swarm Optimization* (PSO), *Ant Colony Optimization* etc, expuestos a lo largo de la presente Tesis Doctoral. Es destacable que los sistemas evolutivos de corte artificial se corresponden con un

gran aliado en lo referido a entender qué procesos de *mapping genotipo-fenotipo* son frecuentes, ayudando en la creación de sistemas artificiales evolutivos.

En ACO [Dorigo, 2004] un conjunto de hormigas artificiales construyen la solución al problema de optimización considerado, e intercambian información basada en la calidad de estas soluciones mediante una variación estigmérgica, proceso basado en el realmente utilizado por parte de los formícidos del mundo real.

Es reseñable que un largo número de distintos algoritmos han sido desarrollados para encontrar soluciones sub-óptimas para el Problema del Viajante (TSP) en un tiempo polinomial, tal y como puede estudiarse en *Nearest Neighbor*, *Greedy*, *Insertion Heuristics*, *Christofides* y $\frac{3}{2}$ -*Opt*. Las metaheurísticas indeterministas como *Simulated Annealing* (SA) [Wang, 2009] [Meer, 2007], Tabu Search [Grendeau, 1998], Genetic Algorithms [Liu, 2009] y PSO [Shi, 2007] han sido a su vez aplicadas, otorgando mejores resultados que los previamente mencionados por los algoritmos puramente deterministas. Existen eficientes algoritmos para la resolución del anteriormente citado TSP, pero la investigación continúa en diferentes algoritmos metaheurísticos. [Rego, 2011]

ACO constituye otra metaheurística exitosamente usada por el TSP [Kollin, 2017], y su efectividad ha sido mejorada utilizando diversos tipos de hibridación, como la adición de agentes locales de búsqueda combinando el propio ACO con GA [Lee, 2008] [Moleshi, 2014], Differential Evolution (DE) [Xiao, 2011] o Kangaroo Algorithm (KA) [Xiang, 2015]. Tales métodos de hibridación tienen, normalmente, una penalización adicional en términos de complejidad en la implementación y, por ende, como es típico, de tiempo. Es importante destacar que se ha demostrado que, si bien añadir agentes de búsqueda locales es un buen enfoque en un gran número de casos, este factor puede evitar que ACO encuentre una solución óptima [Neumann, 2008]. Una muy interesante hibridación de ACO puede estudiarse en [Gan, 2010], donde ciertas hormigas especiales, que persiguen el espacio de búsqueda de una manera sistemática, se añaden al algoritmo. Los sistemas basados en un enfoque multi-colonia [Jovanovic, 2010] también constituyen una interesante alternativa cuando se trata de ACO, así como las variaciones del ACO básico como la basada en elitismo, ranking y min-max (MMAS) para mejorar el rendimiento del TSP [Stutzle, 1999] de una forma más natural, con distintos enfoques en cuanto a las feromonas utilizadas. Adviértase que todas estas variaciones presentan un para nada desdeñable problema, y es que pueden quedarse atrapadas en un óptimo local a tenor del hecho de que incrementan su eficiencia haciendo su búsqueda más ávida (*greedy*) mediante la intensificación de la búsqueda cerca de la mejor solución encontrada. El algoritmo MMAS [Stutzle, 2000] intenta solucionar este problema enlazando los valores que puede tomar la feromona en un rango dado, reseteando el rastro de las mismas si no se encuentran mejores soluciones en un amplio número de iteraciones. La técnica de *Minimum Pheromone Threshold*

Strategy (MPTS) es otro enfoque destacable para evitar el estancamiento temprano aplicado a problemas de asignación cuadráticos. [Wong, 2009]. MPTS añade un valor de umbral (*threshold*) mínimo, y si una feromona cae por debajo del mismo, se establecerá su valor como el valor máximo permitido para el rastro de feromonas. Por consiguiente, MPTS evita la reinicialización del rastro de feromonas, como ocurre en MMAS, y explora el espacio de búsqueda de manera más sistemática. Otro enfoque especialmente interesante es la introducción de la sensibilidad de feromonas variable entre una población de formícidos [Pintea, 2012].

Por ende, y englobando los numerosos factores y estudios anteriormente expuestos, queda demostrado que, desde su concepción en 1991, los sistemas basados en el Paradigma Ant Colony han sido ampliamente utilizados en problemas de optimización principalmente centrados en *path-finding optimization*, como el problema NP Completo *Travelling Salesman Problem* (TSP o Problema del Viajante):

Figura LVII: ACO resolviendo TSP al hallar una ruta mínima entre los puntos de interés

Adviértase que una de las principales ventajas de Ant Colony reside en la elección del borde (*edge*) a explorar, definido gracias al uso de los *pheromone trails* exhaustivamente descritos en el epígrafe dedicado a tal paradigma en la presente Tesis Doctoral.

Englobando ambos factores, se propone el uso de *Ant Colony Optimization* (ACO) para explorar una gramática *Backus-Naur Form* (BNF), cuyos elementos se corresponderán con soluciones a un problema dado. Modelos similares, que no aplican ACO, se han utilizado con frecuencia para resolver problemas de optimización, así como para generar automáticamente problemas tales como *Grammatical Swarm* (basado en *Particle Swarm Optimization*) y *Grammatical Evolution* (basado en *Genetic Algorithms*), ambos detallados minuciosamente en el presente documento.

El Algoritmo ACORD, fruto de la investigación de esta Tesis Doctoral, que ha valido la publicación de [de Mingo et al, 2019] y que se presentará a lo largo del presente epígrafe, abre una nueva vía para una novedosa rama de investigación en el espectro de *Swarm Intelligence* que hasta ahora ha sido prácticamente

inexistente, utilizando ACO para generar soluciones a un problema dado descrito por una gramática BNF, disfrutando de la ventaja del proceso de *mapping* genotipo-fenotipo, descrito en *Grammatical Evolution*. En este caso, tal operación de *mapping* se llevará a cabo basada en una concentración de feromonas para cada regla de producción, recuperando el proceso de *stigmergia* que aglutina la idiosincrasia de ACO. Las características anteriormente citadas, junto con el resto de sus homólogas en el ámbito de ACORD, serán desmenuzadas a lo largo de los siguientes epígrafes.

3.1.1 Definición y Características

Alcanzado este punto de la Tesis Doctoral, es innegablemente evidente que los algoritmos circunscritos a ACO han demostrado ser especialmente útiles alternativas para solucionar problemas de *routing*. El algoritmo ACORD se aprovecha, precisamente, de esta característica para encontrar la solución a un problema de optimización, definido por una función de *fitness* en la que los mejores individuos pertenecen al lenguaje generado por una gramática BNF. Recapitulando, dicho algoritmo utiliza la siguiente terna:

1. Una gramática BNF que representa posibles soluciones del problema planteado.
2. Una función de *fitness* para evaluar tales elementos. (Adviértase el nexo con los paradigmas *Grammatical Evolution* y *Genetic Algorithms*, entre otros.)
3. Una estructura algorítmica basada en el algoritmo clásico *Ant Colony Optimization*.

Una vez definida la terna que se corresponde con la piedra angular de ACORD, se está en situación de presentar el algoritmo *per-se*:

Algorithm Pseudo-code for the proposed Ant Colony algorithm.

```
1: Initialize
2: while stopping criterion not satisfied do
3:   Position each ant in a starting node (first BNF grammar symbol)
4:   Set such symbol as the current solution for every ant
5:   repeat
6:     for Each ant do
7:       Choose next node by applying a BNF rule to the
         left-most non-terminal symbol from the current
         solution
8:     end for
9:   until every ant has built a solution (avoiding large recursive
     derivations)
10:  update best solution
11:  Apply offline pheromone update
12: end while
```

Algoritmo 1: Ant Colony Algorithm for BNF Grammars [de Mingo et al, 2019]

Por tanto, nótese cómo la gramática se utiliza en un proceso incremental que pretende construir una solución mediante la aplicación de las reglas de producción, comenzando desde el símbolo inicial de la gramática. Por otro lado, comenzando por el lado izquierdo del genoma, las hormigas se utilizan para seleccionar las reglas apropiadas para el carácter no-terminal situado más a la izquierda, desarrollando la solución a partir de la gramática BNF, hasta que una de las siguientes casuísticas aparezca:

- Una solución completa sea generada: Este escenario ocurre siempre y cuando todos los caracteres no-terminales de la expresión siendo transformada (*mapped*) se transformen en elementos pertenecientes al conjunto de caracteres terminal de la gramática BNF.
- El umbral (*threshold*) del número de *wrapping events* sea alcanzado y el individuo no esté completamente transformado: En este escenario opuesto, el proceso de mapping se detiene, y se asigna al individuo el menor valor de *fitness* posible, a fin de evitar su propagación en futuros genotipos.

A fin de ilustrar la concentración en cada transición dada por una regla, se presentan las siguientes figuras, primera y segunda iteración, respectivamente:

Figura LVIII: Primera iteración de ACORD [de Mingo et al, 2019]

Adviértase que en esta primera iteración, representada en la parte izquierda de la figura, los nodos azules representan el camino seguido por la hormiga, siendo generados si no están presentes. (Inicialmente, el grafo está vacío, por lo que los nodos azules se generan.) Por otro lado, nótese cómo la concentración inicial de feromonas se establece en 0.5 en un primer momento. Seguidamente, en la fase de actualización de feromonas, sólo los bordes (*edges*) por los que la hormiga ha viajado serán modificados de manera acorde a la función de *fitness*. Los nodos blancos simplemente se muestran para interpretar el grafo de derivación, pero no son generados. La siguiente iteración se muestra en la figura a continuación:

Figura LIX: Segunda iteración de ACORD [de Mingo et al, 2019]

Como puede observarse en esta segunda iteración (izquierda), los nodos coloreados en azul representan de nuevo el camino seguido por la hormiga, mientras que sus homólogos en azul gradiente y amarillo son nodos existentes por los que la hormiga anterior ha transitado (derecha). En la fase de actualización de feromonas, únicamente el borde por el que la hormiga ha viajado es modificado en base a las directrices de la función de *fitness*.

Es reseñable que, para las dos figuras anteriores, se muestran dos iteraciones del algoritmo que tienen en cuenta la siguiente gramática, a modo de ejemplo, que representa números binarios y una función de *fitness* (irrelevante en el ejemplo) que evalúa un número binario dado:

$$\begin{aligned}
 \langle leds \rangle & ::= \langle value \rangle \langle leds \rangle \\
 & \quad | \quad \langle value \rangle \\
 \langle value \rangle & ::= '0' \\
 & \quad | \quad '1'
 \end{aligned}$$

A continuación se presenta la fórmula a la que se recurre en la fase de actualización de hormonas:

$$\Delta\tau_{xy}^k = \begin{cases} QL_k & \text{if ant } k \text{ uses edge } xy \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

Donde L_k es el *fitness* de la solución generada por la hormiga (siendo 0 si el individuo no está completamente transformado, como se especifica en la doble condición expuesta anteriormente en el presente epígrafe) y Q es una constante.

A título ilustrativo, se presenta un ejemplo de una gramática BNF (parte superior de la figura) y el resultado de aplicar la secuencia de reglas al elemento no-terminal `<sentence>`. (*Random Rule Derivation Scheme*, RRDS, con un máximo -criterio de parada- de reglas aplicadas de 15.):

```

<sentence> ::= <subject> <predicate>
<subject>  ::= <article> <noun>
<predicate> ::= <verb> <directobject>
<directobject> ::= <article> <noun>
<article> ::= THE | A
<noun>    ::= MAN | DOG
<verb>    ::= BITES | PETS

```

Result: TRUE (Number of applied rules: 9)		
A DOG PETS THE DOG		
<sentence>	::=	<subject> <predicate>
<subject>	::=	<article> <noun>
<article>	::=	A
<noun>	::=	DOG
<predicate>	::=	<verb> <directobject>
<verb>	::=	PETS
<directobject>	::=	<article> <noun>
<article>	::=	THE
<noun>	::=	DOG

Figura LX: Gramática de entrada y resultado obtenido con un RRDS=15 [de Mingo et al, 2019]

Por otro lado, y en la misma línea a fin de ilustrar de manera más completa el comportamiento del algoritmo, se presenta un ejemplo de otra gramática BNF y el resultado obtenido para el mismo no-terminal, manteniendo un RRDS de 15:


```

<sentence> ::= if <expr> return <expr>
              else return <value>
<expr> ::= ( <expr> <op> <expr> ) | <value>
<op> ::= + | - | * | /
<value> ::= X | Y | Z

```

Result: TRUE (Number of applied rules: 14)		
if Y return (Z * (Z - Z)) else return Y		
<sentence>	::=	if <expr> return <expr> else return <value>
<expr>	::=	<value>
<value>	::=	Y
<expr>	::=	(<expr> <op> <expr>)
<expr>	::=	<value>
<value>	::=	Z
<op>	::=	*
<expr>	::=	(<expr> <op> <expr>)
<expr>	::=	<value>
<value>	::=	Z
<op>	::=	-
<expr>	::=	<value>
<value>	::=	Z
<value>	::=	Y

Figura LXI: Gramática de entrada (II) y resultado obtenido con un RRDS=15 [de Mingo et al, 2019]

A continuación puede encontrarse la implementación del algoritmo ACORD en el lenguaje de programación R. Adviértase que las partes menos significativas del programa se han obviado por razones prácticas:


```

ReadBNFFile <- function(filename) {
  con=file(filename, open="r")
  lines=readLines(con)
  close(con)
  rule_list = list()
  for (l in lines) {
    l = trim_space(l)
    gram_line = strsplit(l, "::~=")[[1]]
    if (length(gram_line) > 1) {
      rules = strsplit(trim_space(gram_line[2]), "|", fixed=TRUE)
      for (j in seq_along(rules[[1]])) rules[[1]][[j]] = trim_space(rules[[1]][[j]])
      i = length(rule_list) + 1
      rule_list[[i]] = list()
      rule_list[[i]][[1]] = trim_space(gram_line[[1]])
      rule_list[[i]][[2]] = as.list(rules[[1]])
    }
  }
  return (rule_list)
}

trim_brackets <- function (x) gsub("^<+|>+$", "", x)
trim_space <- function (x) gsub("^\\s+|\\s+$", "", x)

findRule <- function(grammar, non_terminal) {
  idx <- grep(non_terminal, lapply(grammar, function(x) x[[1]]))
  if (length(idx) == 0) return (NULL)
  grammar[idx][[1]][[2]]
}

apply_rule <- function (expression, rule, consequent, first_NT) {
  expr <- list()
  idx <- grep(first_NT, expression)[1]
  res <- as.list(strsplit(trim_space(rule[[consequent]]), " ", fixed=TRUE)[[1]])
  if (idx > 1) expr <- append(expr, expression[1:(idx-1)])
  expr <- append(expr, res)
  if (idx < length(expression)) expr <- append(expr, expression[(idx+1):length(expression)])
  expr
}

iterate_rule <- function(expression, g) {
  stop <- TRUE
  solution <- FALSE
  derivationlist <- list()
  first_NT <- expression[grep ("<[:alnum:]]>", expression)[1]]
  if (is.null(first_NT[[1]])) ##---- non_terminal NOT found
    solution <- TRUE
  else {
    rule <- findRule(g, first_NT)
    if (is.null(rule)) ##---- rule NOT found for expression
      stop <- TRUE
    else {
      idx <- sample(1:length(rule), 1)
      expression <- apply_rule(expression, rule, idx, first_NT)
      datos <- data.frame(idx, first_NT, rule[idx] )
      colnames(datos) <- c('Idx', 'Antecedent', 'Consequent')
      derivationlist <- append(derivationlist, list(datos))
      stop = FALSE
    }
  }
  list(expression, stop, solution, derivationlist)
}

```

A fin de completar, a modo ilustrativo, el código anterior, se introduce a continuación la función en R de procesamiento de gramáticas aleatorias:

```

source("Utils.R")

processGrammarRandom <- function(input_file, max_iter) {
  g <- ReadBNFFile(input_file)
  expression <- (g[[1]][1])
  expression_list <- list(paste(unlist(expression), collapse=" "))

  rules_applications <- 0
  solution <- FALSE

  derivation_list <- list()
  repeat {
    resultado <- iterate_rule(expression, g)
    if (resultado[[2]] == FALSE)
      rules_applications <- rules_applications + 1
    derivation_list <- append(derivation_list, resultado[[4]])
    expression <- resultado[[1]]
    if ((resultado[[2]] == TRUE) || rules_applications >= max_iter) {
      solution <- resultado[[3]]
      break
    }
    expression_list <- append(expression_list, paste(unlist(expression), collapse=" "))
  }
  list(resultado=resultado, rules_applications=rules_applications, expression_list=expression_list, derivation
    _list=derivation_list, expression=unlist(expression))
}

```

Definido por tanto ACORD, se está en posesión de examinar su comportamiento, misión principal del siguiente epígrafe.

3.1.2 Resultados con Santa Fe Trail

Dado que ACORD representa un híbrido más puramente teórico que FA-IFA, se interpreta que sus experimentos son más cercanos al campo de la *metodología* que al de los *resultados*, por lo que se presentan en el presente epígrafe los frutos obtenidos de aplicar ACORD al problema *Santa Fe Trail*, recurrente en la presente Tesis Doctoral. (Es especialmente reseñable destacar que ACORD es genérico y puede solucionar cualquier camino, no estando únicamente restringido a *Santa Fe Trail*, que se usa a modo de ejemplo, por lo que puede utilizarse para caminos diferentes.) No obstante, los resultados numéricos de ACORD serán mostrados y discutidos en este epígrafe.

El problema *Santa Fe Trail* ha sido, desde su concepción en 1992, un problema extensivamente usado para investigar, probar y evaluar sistemas de computación evolucionaria y diversos métodos a lo largo de las pasadas dos décadas. Sin embargo, no existe literatura relacionada con sus estructuras programáticas, que son sistemáticamente usadas para la mejora de su *fitness*, la geometría de dichas estructuras y las dinámicas que adoptan durante la optimización. Un diagrama de *Santa Fe Trail* puede observarse en la siguiente figura, aprovechando la ocasión para recordar que el objetivo del problema es evolucionar un programa que pueda navegar este rastro, encontrando el sustento (denotado por los cuadrados negros) mientras se utiliza una cantidad concreta de energía:

Figura LXII: Diagrama asociado al problema *Santa Fe Trail* en un *grid* toroidal de 32x32 celdas

Se destaca que el rastro de alimento está compuesto por 144 celdas (de las cuales 89 contienen realmente comida, mientras que las 55 restantes son huecas). Existen 10 giros a la izquierda y 11 a la derecha en el rastro. Es de extrema importancia reseñar que ACORD es capaz de adaptarse a un problema real o camino cambiante gracias a ACO, habiéndose elegido el camino asociado a *Santa Fe Trail* dada su alta frecuencia de aparición en artículos científicos, como se verá seguidamente. No obstante, ACORD no sólo puede resolver el camino *Santa Fe Trail*, sino que puede resolver cualquier camino propuesto, dado su carácter genérico.

Tal y como puede estudiarse en la literatura asociada, las cantidades de energía utilizadas en la solución de este problema varían ampliamente, a saber, en unidades energéticas: 400 en [Kushchu, 2002], 600 en [Oplatkova, 2006] y 615 en [O'Neill, 2001] y [Hugosson, 2007]. En la presente Tesis Doctoral, se utilizará el valor más común que arroja el estudio de la literatura relacionada; 600 unidades energéticas.

Por tanto, el objetivo consiste en encontrar una solución, entendiendo como tal un programa que pueda encontrar toda la comida disponible sin caer exhausto, es

decir, sin utilizar todas las unidades energéticas que tiene a su disposición. Para este problema en concreto, las hormigas actúan como dichos programas.

A pesar de que las directivas a respetar son simples, como se verá a continuación, la necesidad de optimizar todas en cada movimiento hacen crecer la complejidad del problema de forma exponencial. A saber, las hormigas comenzarán en la esquina superior izquierda encarando la sección este de la figura. Una hormiga puede girar a la izquierda o derecha o moverse hacia delante un *step* (paso) en la dirección que está encarando. Además, la hormiga puede sentir si existe comida delante en la dirección que tiene delante. Cada actividad reduce la cantidad total de energía de la hormiga en una unidad.

En ocasiones, *Santa Fe Ant Problem* se utiliza como un problema representativo, usualmente en una *suite* de problemas similares. Son casos comunes, precisamente para este uso, los *benchmarks* de rendimiento para mostrar la viabilidad de meta-heurísticas o nuevos métodos. El problema ha sido utilizado de forma similar para demostrar cómo de factible es la Programación Analítica [Oplatkova, 2006], la Programación Genética Cartesiana, la Evolución Gramatical y la Evolución Gramática Constitutiva [Georgiou, 2011]. Además, *Santa Fe Ant Problem* ha sido utilizado para estudiar la habilidad de Generalización [Kushchu, 2002] y el Coeficiente de Inclinación Negativo [Poli, 2007].

Por otro lado, el problema en cuestión ha sido estudiado por parte de otras metaheurísticas, como los Tipos de Crossover [O'Neill, 1999], el concepto Degeneracy [Ryan, 1998] y su parte opuesta; Wrapping [O'Neill, 1999] en una proporción de individuos inválidos, estudiando su variedad genotípica y la frecuencia acumulativa de éxito. En [O'Neill, 2001], el efecto de los cambios en la gramática estándar y su sesgo mediante la adición de una nueva gramática es investigado con un factor *degeneracy* y sin el mismo. Por otro lado, en [Langdon, 1998] puede apreciarse cómo el problema se redefine, de manera que la hormiga está obligada a traspasar el rastro en aproximadamente el orden correcto. Se demuestra en este último estudio cómo el rendimiento mejora mediante el uso de un sistema de programación genética simple, sin restricciones de tamaño ni profundidad.

Es destacable que el análisis panorámico de la estructura, su localización y la búsqueda a llevar a cabo son factores que no necesariamente deben restringirse, tal y como puede apreciarse en [Rothlauf, 2005], donde se examina la localización de la representación binaria utilizada de manera clásica en los Algoritmos Genéticos. Dicho estudio pone de manifiesto un factor tan sutil como crucial, y es que el operador de mutación binario genotípico produce cambios no locales en el fenotipo. Por ende, se propone una investigación más minuciosa partiendo de dicho estudio, a fin de encontrar otras representaciones y operadores de mutación que produzcan una localidad mayor.

Tal localidad en Grammatical Evolution es también objeto de estudio de [Hugosson, 2007], donde se investiga el impacto de cuatro operadores de mutación en el rendimiento, utilizando representación binaria y en Código Gray, descrito este último precisamente en esta Tesis Doctoral. Las combinaciones de representaciones y operadores de mutación son examinados utilizando tres problemas a modo de *benchmark*, incluyendo *Santa Fe Trail*. Las representaciones alternativas no resultan en una localidad mayor o en un rendimiento mayor, obligando a preguntarse si realmente una mayor localidad en *Grammatical Evolution* mejora, en efecto, el rendimiento. Por ejemplo, el BNF para *Santa Fe* es el siguiente:

```
N = {<code>, <line>, <if-statement>, <op>,
    <expr>}
T = {left(), right(), move(), food_ahead(),
    else, if, {, }, (, ), ;}
S = <code> | <expr>
```

Figura LXIII: Gramática BNF para Santa FE para estudiar el impacto de la localidad en GE

Es reseñable que el símbolo inicial puede ser `<code>` (como ocurre en la definición gramatical de O’Neill) o `<expr>` (en el caso de Koza), mientras que `P` puede ser representado como, respectivamente para cada estudio como:

BNF-O’Neill grammar definition

```
<code> :: = <line> | <code><line>
<line> :: = <if-statement> | <op>
<if-statement> :: = if (food_ahead()) {
                    <line>
                    } else {
                    <line>
                    }
<op> :: = left(); | right(); | move();
```

BNF-Koza grammar definition

```

<expr> ::= <line> | <expr><line>
<line>  ::= <if-statement> | <op>
<if-statement> ::= if(food_ahead()) {
                        <expr>
                    } else {
                        <expr>
                    }
<op> ::= left(); | right(); | move();

```

Alcanzado este punto, se está en situación de describir el apartado más puramente numérico de los resultados de *Santa Fe Trail* tras la aplicación de ACORD, en los que el algoritmo ha sido objeto de funciones de benchmarking con el algoritmo *Grammatical Evolution* (GE) como competidor directo. Dos benchmarks distintos han sido ejecutados en el escenario en el que GE usa la definición de gramática estándar para el problema *Santa Fe Trail*, que define un espacio de búsqueda sesgado, que no es semánticamente equivalente al del problema original. En cada serie de experimentos, se han ejecutado cinco escenarios distintos, consistentes en cien ejecuciones evolutivas cada uno. Un total de quinientas ejecuciones evolutivas han sido, por otro lado, llevadas a cabo para cada algoritmo. La siguiente tabla muestra los datos del experimento llevado a cabo:

Objective	Find a computer program in the NetLogo programming language to control an artificial ant so that it can find all 89 pieces of food located on the Santa Fe Trail.
Terminal Operators	turn-left, turn-right, move, food-ahead.
Raw Fitness	Number of pieces of food picked up before the ant times out with 650 operations
Evolutionary Algorithm	Steady-State Genetic Algorithm, Generation Gap = 0.9 Selection Mechanism: Roulette-Wheel Selection
Initial Population	Randomly created with the following restrictions: Min. Codons = 15 and Max. Codons = 25
Parameters	Population Size = 500, Maximum Generations = 50 Mutation Prob. = 0.01, Crossover Prob. = 0.9 Codon Size = 8, Wraps Limit = 10

Tabla I: Tabla de *inputs* para el experimento contra el Algoritmo ACORD

A continuación se presenta la definición de gramática BNF utilizada para este experimento, donde destaca que el fenotipo de cada gen constitutivo (*) se añade como una regla de producción en el operador no terminal <op>:

```
N = {behaviour, op}
T = {turn-left, turn-right, move, ifelse, food-ahead, [, ]}
S = behaviour
P:
<behaviour> ::= ifelse food-ahead [ <op> ] [ <op> <behaviour> ]
              | ifelse food-ahead [ <op> ][ <op> ]
              | <op>
<op> ::= turn-left | turn-right | move
        | ... {Constituent Genes Phenotypes}(*)
```

Sin más dilación, se adelanta que el Algoritmo ACORD tiene éxito en la empresa de encontrar la solución en el problema Santa Fe Trail con ratios de éxito colosales, que van del 85% al 94% con un ratio de éxito medio del 90%. El código NetLogo para la mejor solución puede estudiarse a continuación:

```
ifelse food-ahead
  [move]
  [turn-right
    ifelse food-ahead
      [ifelse food-ahead
        [move move] [move]
        ifelse food-ahead
          [move move] [move]
      ]
      [turn-left]
    move
  ]
[ifelse food-ahead
  [move] [turn-right]
  ifelse food-ahead
    [turn-left]
    [turn-right
      ifelse food-ahead
        [move]
        [ifelse food-ahead
          [move] [turn-right]
        ]
    ]
  move ]]]]
```

Una vez ofrecido al lector el resultado del experimento, se presenta la terna siguiente de tablas [de Mingo et al, 2019], que muestran los resultados detallados del mismo, donde la columna *Best* muestra el mejor valor de los cinco experimentos, *Runs* el número de ejecuciones evolutivas llevadas a cabo en cada experimento, *Steps* el número de pasos necesario para encontrar la mejor solución en cada experimento, *Success* cuántas ejecuciones evolutivas (en porcentaje) fueron necesarias para hallar una solución y *Average Success*, adscrito al ratio de éxito medio de los cinco experimentos:

	Exp1	Exp2	Exp3	Exp4	Exp5	Best
Runs	100	100	100	100	100	100
Steps	393	375	393	377	337	337
Success	85%	93%	89%	94%	87%	94%
Average Success	90%					

Tabla II: Resultados experimentales para *Santa Fe Trail* aplicado al Algoritmo ACORD

	Exp1	Exp2	Exp3	Exp4	Exp5	Best
Runs	100	100	100	100	100	100
Steps	419	507	415	541	479	415
Success	8%	11%	10%	6%	13%	13%
Average Success	10%					

Tabla III: Resultados experimentales para *Santa Fe Trail* aplicado a *Grammatical Evolution* usando BNF-Koza

	Exp1	Exp2	Exp3	Exp4	Exp5	Best
Runs	100	100	100	100	100	100
Steps	609	609	607	609	607	607
Success	80%	76%	75%	81%	74%	81%
Average Success	78%					

Tabla IV: Resultados experimentales para *Santa Fe Trail* aplicado a *Grammatical Evolution* usando BNF-O'Neill

Los resultados experimentales muestran, por ende, que el Algoritmo ACORD presenta un rendimiento que es varias veces mayor al de *Grammatical Evolution* en términos de ratio de éxito independientemente del tipo de gramática BNF usada. Este pobre rendimiento de GE utilizando BNF-Koza puede explicarse si se tiene en cuenta que esta definición de gramática define un espacio de búsqueda semánticamente equivalente al del problema original, que es hiperbólico, haciendo difícil que GE encuentre una solución. Por otro lado, BNF-O'Neill define un espacio de búsqueda mucho menor, y sesga la misma a áreas donde la

solución puede encontrarse fácilmente, es decir, con un ratio de éxito mayor, pero al coste de excluir otras áreas donde programas más eficientes podrían existir. (En otras palabras, usando menos *Steps*.) Esta es la razón por la que GE con BNF-O'Neill no encuentra soluciones en menos de 607 pasos o *steps*. A fin de reforzar este último razonamiento, es digno de mención que se realizaron cinco experimentos en cien ejecuciones evolutivas, estableciendo un número límite de *steps* de 606. En estos experimentos el ratio de éxito de GE fue del 0%, arrojando una evidencia experimental y empírica de que GE con BNF-O'Neill es incapaz de encontrar soluciones en menos de 607 pasos. Es altamente probable que la amplia distancia de BNF-Koza con respecto a sus competidores se deba a la inexistencia de un umbral de control (*threshold*) a partir el cual se aborten evaluaciones abocadas al fracaso, y es que es especialmente destacable evidenciar que, en ciertas ocasiones, la mezcla de Genetic Algorithm con Grammatical Evolution es ciertamente problemática. Esta perniciosa característica proviene, principalmente, del amplio coste computacional que conlleva aplicar un GA a una gramática que genere un árbol de altura suficiente para ciertos *inputs*. Entendiendo como individuos pertenecientes al genotipo del GA las posibles (y numerosas) variaciones que pueden acontecer al evaluar la gramática en un caso concreto, debe recordarse que GA es especialmente intensivo en cálculo, puesto que para hallar el fenotipo final ejecutará numerosas permutaciones, variaciones, recombinaciones, cruzamientos y otros operadores genéticos sobre el genotipo. Por esta razón, la complejidad (y tiempo de ejecución) al combinar GA y GE podría crecer de manera exponencial siempre y cuando no se controle de alguna manera, tal y como se lleva a cabo con el valor de *threshold* en el caso de ACORD. Así, siempre y cuando el número de evaluaciones sea mayor a un límite aceptable, se abortará la generación de la solución, a la vez que se asigna un bajo valor de *fitness*.

Es destacable, en una línea de investigación paralela, que [Georgiou, 2011] intenta resolver el problema *Santa Fe Trail* recurriendo a una búsqueda aleatoria como el mecanismo de búsqueda de GE en vez del *steady-state* estándar de los Algoritmos Genéticos. Los resultados de dicho experimento demuestran que, al usar el espacio de búsqueda definido por la definición de gramática de BNF-O'Neill, GE con una búsqueda aleatoria tiene un ratio de éxito del 50% en la búsqueda de la solución. En contrapunto, el ratio de éxito de GE con una búsqueda aleatoria usando el espacio de búsqueda definido por la gramática BNF-Koza es de sólo el 1.4%. Estos resultados refuerzan, más aún si cabe, el hecho de que la gramática BNF-O'Neill establece un espacio de búsqueda distinto al propuesto por BNF-Koza, donde las soluciones pueden encontrarse de una manera mucho menos traumática en términos de coste computacional y tiempo.

A fin de completar la información anterior, la siguiente figura muestra el valor de *fitness* contra el número de iteración en *Santa Fe Trail*. En azul oscuro puede

observarse el mejor *fitness*, por otro lado el *fitness* medio se muestra en azul royal y el *fitness* de varianza, por último, en azul *powder*:

Figura LXIV: *Fitness vs # de iteración en Santa Fe Trail problem* usando ACORD con la gramática Koza [de Mingo et al, 2019]

A modo de conclusión se destaca, a tenor de los resultados, que este algoritmo abre una nueva vía de investigación en el ámbito de la *Swarm Intelligence*. Esta vía, inexistente hasta el momento, utiliza algoritmos basados en *Ant Colony* para resolver problemas descritos en una gramática. Es reseñable que el algoritmo propuesto es capaz, a través de su capacidad de cambiar de comportamiento *ad-hoc*, de focalizar la búsqueda en áreas más útiles mediante el decrecimiento del espacio de búsqueda real sin utilizar, en ningún momento, información de dominio o del problema en cuestión. Por otro lado, tampoco necesita cambiar el espacio de búsqueda semántico original en ningún momento, factor que contribuye a su alta velocidad que, como se ha especificado previamente, es apabullante frente a GE. Es fundamental recordar que, además, ACORD es genérico y resuelve cualquier camino, no únicamente *Santa Fe Trail*. En otras palabras, y gracias a *Ant Colony Optimization*, ACORD es capaz de adaptarse a problemas del mundo real o caminos cambiantes.

Los resultados experimentales presentados a lo largo del presente epígrafe muestran que la técnica basada en *Ant Colony Optimization* superan al algoritmo GE estándar en *Santa Fe Trail problem*, independientemente de que éste último utilice un espacio de búsqueda semánticamente equivalente al del problema GP original, o de si usa el espacio de búsqueda sesgado del sistema de *Standard Grammatical Evolution* (BNF-O'Neil) utilizado en la literatura cuando el paradigma es presentado por O'Neil y Ryan en 2003. Por tanto, se concluye que el algoritmo ACORD propuesto logra mejores resultados que el algoritmo de *Standard Grammatical Evolution* en términos no sólo de eficiencia, sino de efectividad a su vez. Los prometedores resultados de ACORD frente al problema *Santa Fe Trail* parecen ser una invitación directa a pensar si el algoritmo puede superar en rendimiento a GE en otros problemas también.

3.2 Improved Fireworks Algorithm

De manera introductoria, y como se presentará al lector a lo largo del presente epígrafe, es reseñable que *Schwefel* [Laguna et al., 2002], *Rosenbrock* [Rosenbrock, 1960], *Ackley* [Ackley, 1987], *Griewanks* [Molga et al., 2005], *Rastrigin* [Rastrigin, 1974], *Six Hump Camel Back* [Molga et al., 2005], *Goldstein* [Dixon et al., 1978] y *Schaffer* [Laguna et al., 2002] son ejemplos de los conocidos métodos de benchmarking matemático que se utilizarán para medir el incremento de la mejora entre FA e IFA. Además, las distintas implicaciones que la calibración de estos algoritmos con las anteriores funciones de benchmarking presentan serán también analizadas. Persiguiendo la más absoluta precisión en los resultados, se utilizará un amplio elenco de criterios de rendimiento para evaluar ambos algoritmos. Se referencia a [Sfu, 2018] para más información sobre las numerosas soluciones de benchmarking existentes en la literatura. Todo el código perteneciente a *Fireworks Algorithm* (así como a *Improved Fireworks Algorithm*) puede encontrarse en el repositorio *GitHub* dedicado a tal efecto:

<https://github.com/ClemencioMoralesLucas/PhD/tree/master/src/phd/cml/fireworks>

Es vital recordar que, dado que (como se especifica en varias secciones de la presente Tesis Doctoral) FA tiene apenas cuatro años en el momento en el que esta Tesis Doctoral se escribe, la mayoría de trabajos de investigación se centran en el aspecto más puramente conceptual que engloba al algoritmo. Por esta razón, el intento de mejora de FA se antoja como extremadamente interesante.

Retomando el tema que ocupa la presente sección, es importante recordar que los algoritmos de búsqueda y optimización han sido extensivamente utilizados y aplicados en numerosos dominios del ámbito de la computación, utilizando diferentes medidas de *performance* para cada dominio en función de su idoneidad. A fin de mantener tales variaciones, un amplio espectro de funciones de benchmarking para probar FA e IFA serán utilizadas.

Por ejemplo, la función de benchmarking Ackley es extensiva en cálculo, mientras que su homóloga Schaffer es matemáticamente directa en lo referido a la evaluación de su fórmula. Estos límites, cuya sección central será copada con otras funciones de benchmarking tal y como se presentará en el estudio que presenta esta Tesis Doctoral, dotará a la investigación de un conjunto de resultados empíricos y certeros. Las variadas métricas de rendimiento para las funciones de benchmarking permitirán medir FA e IFA en numerosos escenarios, siendo posible, como en todo problema de optimización de idiosincrasia semejante, que los algoritmos se comporten de manera efectiva con respecto a una métrica mientras que sean sub-óptimos en otras. Por el motivo anteriormente citado, es de vital importancia evaluar ambos algoritmos en el mayor conjunto de

pruebas posible. Es por ello que en esta Tesis Doctoral se presentarán los resultados de una comparación empírica a gran escala entre doce funciones de benchmarking seleccionadas con precisión, que serán aplicadas tanto a FA como a IFA en igualdad de condiciones.

A modo de introducción en tal sentido, se destaca que el rendimiento de los algoritmos será evaluado de manera acorde a las funciones de benchmarking *Sphere*, *Schwefel*, *Rosenbrock*, *Ackley*, *Griewanks*, *Rastrigin*, *Penalized*, *Six-Hump Camel Back*, *Goldstein*, *Schaffer*, *Axis Parallel* y *Rotated Hyper-Ellipsoid*.

Alcanzado el presente punto, y sin más dilación, se presenta al lector la mejora de FA, denominada IFA, donde se pormenoriza qué se ha mejorado y cómo. Persiguiendo la finalidad de incrementar el rendimiento algorítmico de FA, se ha llevado a cabo un exhaustivo análisis de dicha entidad, especialmente en términos de *performance* o rendimiento. En tanto en cuanto un análisis en profundidad ha sido presentado al lector en el epígrafe 2.1.1 *Fireworks Algorithm*, se referencia al mismo a tal sección para recordar los conceptos algorítmicos pertinentes, mientras que este epígrafe se centrará en las áreas de mejora propuestas para FA, que desembocan en IFA. A modo de introducción, IFA se verá introducido y desmenuzado a lo largo del presente capítulo.

De manera paralela a FA, descrito a lo largo de la presente Tesis Doctoral, IFA inicializa N fireworks, así como su *fitness* asociado, que se evalúa a fin de determinar la amplitud de la explosión y el número de *sparks* para cada *firework*. Seguidamente, el *firework* explota y genera distintas *sparks* en su espacio local. A fin de asegurar la diversidad y el balance entre la búsqueda local y la global, la amplitud de la explosión y la población de los nuevos *sparks* generados varían entre los distintos *fireworks*. Un *firework* con una mejor función *fitness* puede generar una población mayor de *explosion sparks* en un rango menor, o lo que es lo mismo, con una amplitud de explosión menor. En contraposición, los *fireworks* con un menor valor de *fitness* únicamente pueden generar una población menor en un rango más grande. En otras palabras, y de forma paralela, con una gran amplitud de explosión.

Adviértase que esta técnica permite la existencia de un balance entre *exploración* y *explotación* en el algoritmo, entendiendo como *exploración* su habilidad para explorar varias regiones del espacio de búsqueda a fin de localizar soluciones prometedoras, mientras que la *explotación* se refiere a la capacidad algorítmica para conducir una búsqueda exhaustiva en un área menor, que se reconoce como prometedora en pos de la solución óptima.

Continuando el razonamiento anterior, destaca que la *exploración* se lleva a cabo por los *fireworks* con grandes amplitudes de explosión o menor *fitness*, puesto que disfrutan de la capacidad de escapar de mínimos locales. La *explotación* se

consigue, de forma opuesta, por los *fireworks* con amplitudes de explosión menores o altos valores de *fitness*, puesto que refuerzan la búsqueda local en áreas susceptibles de contener resultados interesantes.

Es importante destacar que un nuevo tipo de *sparks* se generarán basadas en la explosión *Gaussiana* de *Fireworks*. Dichas *sparks* serán seleccionadas de manera aleatoria. Así, se asegurará la diversidad de la población de *fireworks*, por lo que para retener de forma paralela la información del genotipo (realizando un símil con los GA) y propagarse a la siguiente iteración, un subconjunto de los *fireworks* se seleccionará al final del algoritmo, que terminará una vez se alcance el criterio *terminal*. Dicha información se recoge en el siguiente algoritmo:

Algorithm 1 – The general structure of FWA.

```
1: Initialize  $N$  fireworks and set the constant parameters
2: repeat
3:   // Explosion operator
4:   Calculate the explosion amplitudes
5:   Calculate the numbers of explosion sparks
6:   Generate “explosion sparks” (check if out-of-bounds)
7:   Evaluate the fitness
8:   // Gaussian mutation operator
9:   Generate “Gaussian sparks” (check if out-of-bounds)
10:  Evaluate the fitness
11:  // Selection strategy
12:  Select fireworks for the next iteration
13: until termination criterion is met.
```

Algoritmo II: Estructura general de FA

3.2.1 Explosión Gaussiana de Sparks

Tal y como se ha introducido en la presente Tesis Doctoral, la explosión constituye uno de los operadores más fundamentales en el ámbito de FA. Puesto que esta técnica pertenece al espectro de la Computación Natural, toma su inspiración en un fenómeno del mundo real. Entendiendo el enorme espacio de búsqueda/optimización que FA recibe como *input* como un espacio del mundo real, es lícito preguntarse cómo es factible *predecir*, sin ninguna suerte de dato

previo, dónde se esconderán los valores perseguidos. Como se presenta a continuación, en la naturaleza, dichos datos presentan un patrón matemático clásico:

Supóngase ciertas distribuciones probabilísticas cualesquiera. Si se tomaran N muestras de la distribución y se calculara su valor medio, podría denominarse a tal cuantía a_1 . En este momento, podrían tomarse otras N muestras y calcular su media de nuevo, a_2 , repitiendo este proceso una y otra vez hasta lograr un flujo de números, cada uno representando la media aritmética de N muestras separadas, tal que:

$$a_1, a_2, a_3, \dots, a_k$$

Es demostrado [Casella et al., 2001] que, independientemente de la distribución estadística que se tome de inicio (Poisson [Haight, 1967], Binomial [Wadsworth, 1960], Chi-Square [Abramowitz et al., 1983] o cualquier otra), mientras N crece los resultados tenderán a convertirse en una distribución Gaussiana o Normal, cuyo histograma puede estudiarse en la siguiente figura:

Figura LXV: Histograma de la distribución Gaussiana

Más formalmente, en el límite mientras que N tiende al infinito, la distribución de la media aritmética de N muestras de la distribución se distribuye, valga la redundancia, como una distribución Gaussiana. Esta es, precisamente, la idea

que subyace bajo el Teorema Central del Límite [Bauer, 2001], como se verá posteriormente.

El Teorema Central del Límite establece que, siendo $X_1, X_2, \dots, X_n$ variables aleatorias equidistribuidas e independientes tal que la media de $X_i = a$ y la varianza $V(X_i) = b^2$, la serie dada por la suma de las ' n ' variables aleatorias expuestas sigue aproximadamente una distribución Normal (μ, σ) de parámetros:

$$\begin{aligned}\mu &= n * a \\ \sigma &= b * \sqrt{n}\end{aligned}$$

Dicho paralelismo es posible gracias a la propiedad de reproductividad que presenta la distribución Normal, que establece que la variable aleatoria obtenida al sumar dos ó más variables aleatorias normales también sigue una distribución Normal. Precisamente, el nombre de esta distribución viene motivado, como se dejó entrever previamente, por el Teorema Central del Límite, pues se cumple que el suceso seguro o "normal" al estudiar distribuciones reproductivas se corresponde con que, cuando se suma un número de variables que tiende al infinito, se obtiene una distribución Normal de exactitud inversamente proporcional a la distancia al infinito de dicho número.

El Teorema Central del Límite establece que, bajo ciertas condiciones (que normalmente son comunes), la suma de un numeroso elenco de variables aleatorias tendrá, aproximadamente, una Distribución Gaussiana [Johannes, 2003]. De forma más específica, y suponiendo $X_1, \dots, X_n$ como variables aleatorias idénticamente distribuidas, independientes, con la misma distribución arbitraria, media igual a cero y varianza σ^2 , con Z siendo su media escalada por $\sqrt{n}$, entonces:

$$Z = \sqrt{n} \left(\frac{1}{n} \sum_{i=1}^n X_i \right)$$

Por consiguiente y estudiando la fórmula, a medida que la población es mayor (N se incrementa), la distribución probabilística de Z tenderá a ser una Distribución Gaussiana, de media 0 y varianza σ^2 . A continuación se muestra una comparativa de varias funciones de densidad probabilísticas mostrando su convergencia a la Distribución Gaussiana conforme N crece:

Figura LXVI: Conversión de varias distribuciones probabilísticas hacia la Distribución Gaussiana conforme N crece (Teorema Central del Límite)

Es importante destacar que el Teorema Central del Límite puede extenderse a variables X_i que no son independientes o bien no son distribuidas idénticamente si ciertas constantes son supuestas a tenor del grado de independencia y el momento de las distribuciones.

Numerosas entidades, experimentos y estimaciones, en la práctica, presentan sumas de ciertas variables aleatorias en ellos, e incluso más estimadores pueden ser representados como sumas de variables aleatorias a través del uso de las denominadas Funciones de Influencia. El Teorema Central del Límite implica que tales parámetros estadísticos presentarán Distribuciones Gaussianas Asintóticas. A su vez, el teorema implica que ciertas distribuciones pueden aproximarse a la Distribución Gaussiana, por ejemplo:

- Distribución Binomial: $B(n, p)$ es aproximadamente normal con media $n * p$ y varianza $n * p * (1-p)$ si n es grande y p no es cercana a 0 ó 1.
- Distribución *Chi-Squared*: $\chi^2(k)$ es aproximadamente normal con media k y varianza $2k$ si k es grande.

La precisión de estas aproximaciones dependerá, en gran medida, de su propósito y el ratio de convergencia de la distribución origen a la Gaussiana. Es importante reseñar que típicamente tales aproximaciones son menos precisas en las colas de la distribución.

Retomando el tema de apertura del presente epígrafe, y como el Teorema Central del Límite demuestra de forma empírica, puede entreverse cómo la naturaleza *tiende* a la Distribución Gaussiana. Así, en el ámbito contextual del FA, los resultados o puntos de interés encerrados en el vasto espacio de búsqueda del FA también lo harán, por lo que si se seleccionan como puntos de análisis prioritario los que salgan de la Distribución Gaussiana, se podría, teóricamente, mejorar el rendimiento algorítmico.

Por tanto, persiguiendo la mejora del FA, se ampliará la aplicación en el mismo de la Distribución Gaussiana, haciendo que un valor aleatorio salido de la misma se aplique a cada dirección de búsqueda de las *sparks*. En otras palabras, se extraerán de la función $f(x)$, donde μ es la media de la Distribución, σ la desviación estándar y σ^2 la varianza:

$$f(x | \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

De esta forma, se persigue la mejora de la precisión de las *sparks* mediante el aprovechamiento de valores de la Distribución Gaussiana, pues la hipótesis es que estos apunten, de manera más o menos certera, a los puntos de interés. El Algoritmo II demuestra cómo pueden aprovecharse los valores pertenecientes a la Distribución Gaussiana en la instrucción número 3 (*Calculate offset displacement*), cuyo resultado será almacenado en la variable e . Como puede apreciarse especialmente en la instrucción número 5 del algoritmo, dicha variable e será utilizada para variar la posición inicial de la “*Gaussian spark*”, otorgando el matiz diferencial del algoritmo al acotar el área donde podría estar ubicado el punto de interés en cuestión.

Recapitulando los aspectos previamente descritos, se presenta a continuación el algoritmo de explosión de *sparks* en IFA:

Algorithm 2 – Generating one “explosion spark” in EFWA

- 1: Initialize location of “explosion sparks”: $\hat{x}_i = X_i$
 - 2: Set $z^k = \text{round}(\text{rand}(0, 1))$, $k = 1, 2, \dots, d$.
 - 3: **for** each dimension of $\hat{x}_i$, where $z^k == 1$ **do**
 - 4: Calculate offset displacement: $\Delta X^k = A_i \times \text{rand}(-1, 1)$
 - 5: $\hat{x}_{ik} = \hat{x}_{ik} + \Delta X^k$
 - 6: **if** $\hat{x}_{ik}$ is out of bounds **then**
 - 7: randomly map $\hat{x}_{ik}$ to the potential space
 - 8: **end if**
 - 9: **end for**
-

Algoritmo III: Generación de un *explosion spark* en IFA

Tal y como puede apreciarse, cada *firework* explota creando un número diferente de *explosion sparks* en un rango dado por su localización actual. Para cada *explosion spark* s_i de cada *firework* X_i , el algoritmo superior se ejecuta una vez. Una vez explicada la técnica de explosión utilizada, se está en posesión de estudiar el siguiente algoritmo, que detalla cómo se generan las *sparks* Gaussianas en IFA:

Algorithm 3 – Generating one “Gaussian spark” in EFWA

- 1: Initialize the location of the “Gaussian sparks”: $\bar{x}_i = X_i$
 - 2: Set $z^k = \text{round}(\text{rand}(0, 1))$, $k = 1, 2, \dots, d$
 - 3: Calculate offset displacement: $e = \text{Gaussian}(0, 1)$
 - 4: **for** each dimension $\bar{x}_i$, where $z^k == 1$ **do**
 - 5: $\bar{x}_{ik} = \bar{x}_{ik} + (X_{bk} - \bar{x}_{ik}) * e$, where X_b is the position of the best firework found so far.
 - 6: **if** $\bar{x}_{ik}$ is out of bounds **then**
 - 7: $\bar{x}_{ik} = X_{min}^k + \text{rand}(0, 1) * (X_{max}^k - X_{min}^k)$
 - 8: **end if**
 - 9: **end for**
-

Algoritmo IV: Generación de un *Gaussian spark* en IFA

Como puede apreciarse, el algoritmo se ejecuta un número constante, M_g , de ocasiones. Advuértase que M_g representa una constante que controla el número de *Gaussian sparks* disparadas, mientras que $\text{Gaussian}(a, b)$ denota un valor

aleatorio extraído de una distribución normal o Gaussiana con valor a y varianza b . Cada una de estas ocasiones, sobre un *firework* seleccionado al azar X_i .

Un factor clave en el desempeño algorítmico de IFA es la propia amplitud de la explosión, en la que existen dos grandes vertientes:

La primera gran opción, y precisamente la utilizada, *dynFWA*, se corresponde con una estrategia de actualización de las amplitudes de la explosión en función de la calidad de las *sparks*. El algoritmo asociado a dicha estrategia puede apreciarse seguidamente:

Algorithm 4 – *dynFWA* explosion amplitude update strategy

Require: Define:

X_{CF} is the current location of the CF;
 x_b is the best location among all explosion sparks;
 A_{CF}^t is the CF's explosion amplitude in iteration t ;
 C_a is the amplification factor;
 C_r is the reduction factor;

Ensure:

```

1: if  $f(x_b) - f(X_{CF}) < 0$  then
2:    $A_{CF}^t \leftarrow A_{CF}^{t-1} \times C_a$ ;
3: else
4:    $A_{CF}^t \leftarrow A_{CF}^{t-1} \times C_r$ ;
5: end if

```

Algoritmo V: Estrategia de actualización de amplitud de la explosión en *dynFA*

Como puede comprobarse en el algoritmo anterior, siendo x_b el mejor valor de una *spark* recientemente creada en una explosión para todo *firework* perteneciente al genotipo, la amplitud de la explosión se adapta en función de la calidad de las *sparks* generadas, pudiendo existir dos grandes bifurcaciones:

1. Una o más *sparks* han encontrado una mejor posición, siendo posible que una (a) *spark* generada por CF haya encontrado la mejor posición, o que (b) la *spark* generada por un *firework* distinto a CF haya encontrado la mejor posición. Ambos casos indican que el genotipo ha encontrado una posición nueva y prometedora, x_b , que será el CF para la siguiente iteración.

- a. En numerosos casos, x_b será creada por el CF. En tales escenarios, a fin de ampliar la convergencia del algoritmo, la amplitud de la explosión de CF para la siguiente iteración será aumentada con respecto a la actual.
- b. En el resto de casos, un *firework* distinto a CF creará x_b . En dichas ocasiones, x_b se convertirá en el nuevo CF para la siguiente iteración. Dado que la posición de CF cambia, la amplitud de la explosión actual para el CF actual no será efectiva para el nuevo CF. Sin embargo, es posible que x_b se encuentre en un rango cercano al CF previo, puesto que el hecho de que el método de selección aleatoria seleccione varias *sparks* creadas por el CF, significa que dichas *sparks* se localizan en un entorno muy cercano al CF. En tal caso, se aplican las mismas consideraciones que en (a). Si x_b se crea por un *firework* que no está en las proximidades de CF, la amplitud de la explosión puede reiniciarse al valor predefinido. No obstante, y dada la dificultad de definir cuáles son cuantitativamente las “proximidades” anteriormente citadas, no se computa la distancia entre x_b y x_{CF} , confiando en la habilidad de actualización de la amplitud. De manera similar a (a), la amplitud de la explosión se incrementa. Si el nuevo CF no puede mejorar su localización en la siguiente iteración, el nuevo CF es capaz de ajustar su propia amplitud de explosión de manera dinámica.

Adviértase cómo incrementar la amplitud de la explosión puede acelerar la velocidad de convergencia: Asumiendo que la posición actual del CF está lo suficientemente lejos del mínimo local o global. Incrementar la amplitud de la explosión es un enfoque efectivo y directo en pos de incrementar el tamaño de step hacia el óptimo local/global en cada iteración, puesto que permite realizar movimientos más rápidos hacia el óptimo. No obstante, se debe asegurar que el *firework* se aproxima a la posición óptima, bien sea local o global. De hecho, la probabilidad de encontrar una posición con un mejor valor de *fitness* decrece con una amplitud de explosión creciente a tenor del incremento en el espacio de búsqueda, que crece a su vez.

2. Ninguno de los *explosion sparks* de CF ni ningún otro *firework* ha encontrado una posición con mejor valor fitness en comparación a CF. Bajo esta circunstancia, la amplitud de la explosión en CF se reduce a fin de circunscribir la búsqueda a una región más pequeña alrededor de la localización actual, así como para mejorar la capacidad de explotación del CF. La probabilidad de encontrar una posición con un mejor valor de

fitness normalmente se incrementa con el decremento de la amplitud de la explosión.

Por otro lado y finalmente, destaca AFWA en lo referido a la amplitud de la explosión [Zheng et al., 2017]. En este enfoque, la motivación reside en calcular la amplitud más recomendable para el CF sin parámetros pre-seleccionados, haciendo un uso total de la información regional basada en los *sparks* generados alrededor de la posición actual. AFWA pretende encontrar un *spark* específico y utilizar su distancia al mejor individuo (que puede ser x_{CF} o x_b y será seleccionado como CF en la siguiente iteración) como amplitud de la explosión en el siguiente *step*. El *spark* específico que AFWA elige debe respetar las siguientes dos condiciones:

1. Su *fitness* es peor que el asociado al CF actual, que garantiza que el *spark* específico no está en las proximidades de CF.
2. La distancia entre la *spark* específica y el mejor individuo es mínima entre todos los candidatos, a fin de asegurar la convergencia algorítmica.

Tal y como se especifica en lo referido a FA en capítulos previos de la presente Tesis Doctoral, los *explosion sparks* se generan independientemente en cada dimensión, verificando que:

$$||X||_{\infty} = \max_i(|x_i|)$$

La actualización de la amplitud de la explosión en AFWA incluye, a su vez, dos grandes casos:

1. El CF no genera *sparks* con valores de *fitness* menores a los del *firework*. Por ejemplo, el mejor *fitness* de todos los *explosion sparks*, $x_{CF,b}$ es peor que el *fitness* asociado al *firework* actual, x_{CF} , y la amplitud de la explosión será actualizada acorde a la *infinite norm* entre el *firework* actual y la *spark* específica. En ese caso, la amplitud de la explosión en la siguiente iteración será reducida en función del Algoritmo 5, que se presenta a continuación:

Algorithm 5 AFWA explosion amplitude update strategy

Require: Define:

M is explosion sparks number of CF;
 A_{CF}^t is the CF's explosion amplitude in iteration t ;
 $x_{CF,k}$ is the k -th explosion spark of CF;
 $x_{CF,b}$ is the best explosion spark of CF;
 λ is a constant parameter which controls the update rate.

Ensure:

```
1: for  $k = 1$  to  $M$  do
2:   if  $\|x_{CF,k} - x_{CF,b}\|_{\infty} > A_{CF}^t$  and  $f(x_{CF,k}) > f(x_{CF,b})$ 
     then
3:      $A_{CF}^t \leftarrow \|x_{CF,k} - x_{CF,b}\|_{\infty}$ 
4:   end if
5: end for
6:  $A_{CF}^t \leftarrow 0.5 \cdot (A_{CF}^{t-1} + \lambda \cdot A_{CF}^t)$ 
```

Algoritmo VI: Estrategia de actualización de amplitud de la explosión en AFA

2. El $x_{CF,b}$ generado tiene un fitness menor que x_{CF} , y en función del algoritmo anterior será difícil determinar si la amplitud de la explosión será incrementada o decrementada. Generalmente, la amplitud de la explosión se incrementará con gran probabilidad mientras que se reducirá con menor probabilidad. Además, y dado que la *infinite norm* entre la *spark* específica y el *firework* cambia radicalmente, AFWA introduce la siguiente estrategia de suavizado (*smoothing*):

$$(A_{CF}^t \leftarrow 0.5 * (A_{CF}^{t-1} + \lambda * A_{CF}^t))$$

3.2.2 Divide & Conquer, Loop Unrolling y Decremento de la Complejidad Algorítmica

A tenor de que FA es un algoritmo de optimización/búsqueda, existen numerosas secciones del mismo cuya ejecución será iterada, miles (o millones) de veces en función del problema a resolver. Por tal motivo, es de extrema importancia que estos bloques algorítmicos sean analizados con total minuciosidad, puesto que un impacto negativo en el rendimiento de los mismos podría verse multiplicado millones de veces, lastrando el rendimiento algorítmico.

En tal sentido, la sección que más recursos consume en el espectro del algoritmo es la evaluación de la calidad de los sparks. Esta sección no sólo es iterativa, sino que debe realizarse una vez por cada spark lanzada por FA. Dicho comportamiento se traduce en que la mejora de este área es fundamental para lograr el IFA. A continuación se detalla cómo puede llevarse a cabo dicha mejora en términos software (disponible bajo <https://github.com/ClemencioMoralesLucas/PhD/tree/master/src/phd/cml/fireworks>), que contiene la implementación de Fireworks Algorithm e Improved Fireworks Algorithm, así como los pertinentes test unitarios del código fuente desarrollado mediante la técnica *Test Driven Development*):

- La calidad se determina por dos bucles, que representan la matriz del FA, con uno adicional anidado para las sparks. Entonces, otros tres bucles no anidados existen en el método con propósito de evaluación de resultados. En términos de la Notación Bachmann-Landau [Bachmann, 1894], también conocida como *Big O Notation*:

$$O(n) = 2N^2 + 3N$$

- A tenor de [Bachmann, 1894], todo a excepción del término mayor debe eliminarse en la fórmula. Por consiguiente:

$$O(n) = 2N^2$$

- Finalmente, Big O Notation implica eliminar toda constante, dado que su impacto es mínimo. Por tanto:

$$O(n) = N^2$$

Por consiguiente, puede inferirse que la complejidad algorítmica en su sección más pesada es de índole cuadrática. A fin de reducir la complejidad de esta sección se propone en el código un esquema basado en la técnica Divide & Conquer [Brassard et al., 1996].

Es interesante hacer un inciso para describir Divide & Conquer, el cual se corresponde con un paradigma de diseño algorítmico basado en la recursión multi-rama. Esta técnica parte de la reestructuración recursiva de un problema de enjundia, con una solución compleja, en un conjunto de problemas más pequeños (subproblemas) con una solución más simple. Así, llegará un punto en el que los problemas son triviales (fin de la recursión), donde su solución es inmediata. A partir de este punto, la solución a todos los sub-problemas se combina, llegando

a la solución final del problema primigenio [Levitin, 2002]. Es reseñable que, aunque con frecuencia el enfoque algorítmico es recursivo, este también puede ser iterativo.

Es destacable que el paradigma Divide y Vencerás pretende encontrar la solución a un problema mediante tres fases:

1. *Divide*: El problema se escinde en subproblemas del mismo tipo.
2. *Conquer*: Solución (recursiva o iterativa) de dichos subproblemas.
3. *Combine*: Las soluciones de los subproblemas se combinan de manera que el problema principal queda resuelto.

Baste citar como ejemplo el algoritmo de ordenación *Quicksort*, que se apoya en Divide y Vencerás mediante la elección de un elemento pivote en la búsqueda. Una vez seleccionado dicho elemento, el vector de elementos se reorganiza de forma que todos los elementos más pequeños que el pivote se sitúan a su izquierda, mientras que aquellos elementos mayores quedan a la derecha. Finalmente, el algoritmo ordena recursivamente los vectores parciales que quedan a la izquierda y a la derecha del pivote. Puede advertirse, gracias a *Quicksort*, como el problema de ordenación se escinde (*divide*) en dos homólogos más sencillos siguiendo el esquema Divide y Vencerás. Sobre dicho proceso, que genera dos vectores más pequeños, se aplica el siguiente paso (*conquer*), que resuelve recursivamente dichos subproblemas. Finalmente, el vector de elementos menores ordenado a la izquierda del pivote se combina con su vector simétrico de elementos mayores, situado a la derecha del pivote, obteniendo la solución final (*combine*).

Retomando la aplicación de Divide y Vencerás, se destaca que una de las secciones de IFA que más recursos consume a nivel computacional se corresponde con la selección de la mejor posición de las *sparks*. Dicho método, en su implementación clásica para FA, se presenta a continuación:


```

private Spark selectBestLocation() {
    Spark bestSpark = fireworks[0];
    int i, j;
    for (i = 1; i < locationsNumber; i++) {
        if (fireworks[i].getValue(benchmarkFunction) < bestSpark.getValue(benchmarkFunction)) {
            bestSpark = fireworks[i];
        }
    }
    for (i = 0; i < locationsNumber; i++) {
        for (j = 0; j < sparks[i].length; j++) {
            if (sparks[i][j].getValue(benchmarkFunction) < bestSpark.getValue(benchmarkFunction)) {
                bestSpark = sparks[i][j];
            }
        }
    }
    for (i = 0; i < auxSparksNumber; i++) {
        if (auxSparks[i].getValue(benchmarkFunction) < bestSpark.getValue(benchmarkFunction)) {
            bestSpark = auxSparks[i];
        }
    }
    optimumValue = bestSpark.getValue(benchmarkFunction);
    return bestSpark;
}

```

Se aprecia, por tanto, que existen tres grandes bucles *for*, uno de ellos doble, puesto que contiene otro *for* homólogo anidado. Es fundamental destacar que esta operación es clave en el desempeño del algoritmo, puesto que podría ejecutarse millones de veces para un *input* de tamaño suficiente. Por tanto, gracias a la aplicación de Divide y Vencerás, esta importante sección puede simplificarse notoriamente tal y como se presenta a continuación:

```

Spark getBestSparkDivideAndConquer(Spark spark, Spark[] locationsNumber) {
    int n = locationsNumber.length;
    if (n == 1) { //If there is just one spark, it trivially is the best one
        return locationsNumber[0];
    } else {
        if (spark != null) {
            return getBestSparkDivideAndConquer(spark, Arrays.copyOfRange(locationsNumber, 0,
                locationsNumber.length / 2));
        } else {
            return getBestSparkDivideAndConquer(spark, Arrays.copyOfRange(locationsNumber,
                locationsNumber.length / 2, locationsNumber.length));
        }
    }
}

```

Como puede observarse, y de manera similar a como ocurre en *Quicksort*, la aplicación de Divide y Vencerás en IFA desemboca en que el vector que contiene las *sparks* se analizará tras ser escindido en dos mitades. En la primera, se tratarán las *sparks* situadas entre la posición 0 y la sección media del vector, que está ordenado crecientemente en función de la aptitud (*fitness*) de las *sparks*. Por otro lado, las *sparks* con un mayor valor de aptitud se situarán entre `locationsNumber.length / 2` y `locationsNumber.length`. Tras ser evaluadas conforme a la iteración actual, la mejor *spark* será, finalmente, seleccionada. Tal escenario denota el caso base del algoritmo ($n=1$), en el que se infiere que, habiendo una única *spark* en el vector, esta es, trivialmente, la mejor para la iteración actual. (Solución de complejidad $O(n) = N$.)

Además de Divide y Vencerás, destaca la aplicación, en aquellos casos pertinentes, de la técnica de Desenroscado de Bucles (*Loop Unrolling*). Esta técnica de optimización pretende incrementar la velocidad de ejecución de un programa a expensas del tamaño del mismo, término a veces referido como intercambio espacio-temporal (*space-time tradeoff*).

Más en concreto, el Desenroscado de Bucles pretende reducir o eliminar las instrucciones de control dentro de los bucles, tales como el puntero aritmético y las pruebas terminales al final de cada iteración. Además, la técnica reduce las penalizaciones de branch y enmascara latencias provocadas por el retraso que acontece al leer información de la memoria principal. En resumen, la técnica Desenroscado de Bucles persigue la reescritura de los bucles del programa como una secuencia repetida de instrucciones similares independientes. Tal y como el siguiente ejemplo ilustra, también es posible utilizar el carácter par o impar de las posiciones del índice [Ullman, 1977]:

Normal loop	After loop unrolling
<pre> x(1) := 1; For i := 2:9 do x(i) := x(i - 1) * i; print i, x(i); next i; </pre>	<pre> x(1) := 1; x(2) := x(1) * 2; print 2, x(2); x(3) := x(2) * 3; print 3, x(3); x(4) := x(3) * 4; print 4, x(4); ... etc. </pre>

Dicha reescritura ha sido llevada a cabo en todas las partes del programa donde ha sido posible, destacando el siguiente ejemplo, donde primero puede observarse cómo era el bucle inicialmente:

```

private void getExplosionAmplitudeForAllFireworks(double minimumValue, double sumMinDiff, double[] explosionAmplitude) {
    int i;
    for (i = 0; i < locationsNumber; i++) {
        explosionAmplitude[i] = (fireworks[i].getValue(benchmarkFunction) - minimumValue + EPS_VALUE) / (sumMinDiff + EPS_VALUE) * maximumAmplitudeNumber;
    }
}

```

Tras la aplicación de la técnica Desenroscado de Bucles, cada iteración del bucle anterior consta de cinco pasos en vez de uno (de forma previa se asegura que `locationsNumber` contiene un número de posiciones divisible por 5):

```

private void getExplosionAmplitudeForAllFireworks(double minimumValue, double sumMinDiff, double[] explosionAmplitude) {
    int i;
    for (i = 0; i < locationsNumber; i += 5) {
        explosionAmplitude[i] = (fireworks[i].getValue(benchmarkFunction) - minimumValue + EPS_VALUE) / (sumMinDiff + EPS_VALUE) * maximumAmplitudeNumber;
        explosionAmplitude[i + 1] = (fireworks[i + 1].getValue(benchmarkFunction) - minimumValue + EPS_VALUE) / (sumMinDiff + EPS_VALUE) * maximumAmplitudeNumber;
        explosionAmplitude[i + 2] = (fireworks[i + 2].getValue(benchmarkFunction) - minimumValue + EPS_VALUE) / (sumMinDiff + EPS_VALUE) * maximumAmplitudeNumber;
        explosionAmplitude[i + 3] = (fireworks[i + 3].getValue(benchmarkFunction) - minimumValue + EPS_VALUE) / (sumMinDiff + EPS_VALUE) * maximumAmplitudeNumber;
        explosionAmplitude[i + 4] = (fireworks[i + 4].getValue(benchmarkFunction) - minimumValue + EPS_VALUE) / (sumMinDiff + EPS_VALUE) * maximumAmplitudeNumber;
    }
}

```

Como resultado de esta modificación en el caso de ejemplo, el bucle lleva a cabo únicamente una quinta parte de las iteraciones. Por consiguiente, sólo un 20% de los saltos condicionales necesita ejecutarse, lo que desemboca, tras un número de iteraciones lo suficientemente alto, en un decrecimiento significativo de la carga computacional del bucle. Es importante destacar que, para producir un beneficio óptimo e independientemente del lenguaje de programación utilizado, ninguna variable que requiera aritmética de punteros debe definirse en el nuevo código del bucle.

Sin embargo, a pesar de las ventajas que brinda la técnica Desenroscado de Bucles, también existen factores negativos, puesto que el código fuente crece. Estas nuevas líneas deben ser compiladas, probadas y trazadas, además de que el compilador podría necesitar ubicar más registros para almacenar los nuevos operadores pertenecientes a la nueva versión del bucle. Adicionalmente, las variables de control del bucle y el número de operaciones ubicadas en su interior deben ser elegidas cuidadosamente de forma que el resultado es, en efecto, el mismo que el del código original (asumiendo el escenario en el que la técnica se aplica a un código que ya funciona, como es el caso). No obstante, y como se detallará posteriormente en la presente Tesis Doctoral, la aplicación de la técnica en cuestión otorga una notoria ganancia al algoritmo.

En dicha línea, y a fin de lidiar con la posible (e indeseada) mutabilidad del resultado que podría acontecer tras implementar la técnica Desenroscado de Bucles, se ha optado por la implementación de una extensa *suite* de *tests* unitarios. Dichos *tests* se encargan automáticamente de ejecutar el *software* y comprobar que el resultado obtenido es el esperado, por lo que basta con lanzar la *suite* una vez se ha implementado la técnica para poseer indefectiblemente la certeza de que el programa sigue generando los mismos resultados, que son los esperados:

Como resultado, y como se ha destacado anteriormente, la sección adscrita a la evaluación de calidad de sparks en el código, se divide en sub-problemas de trivial solución de manera recursiva, siendo solucionados en tiempo $O(n) = N$ gracias a Divide y Vencerás. Este hecho, en colaboración con la técnica Desenroscado de Bucles (*Loop Unrolling*) [Ullman et al., 1977], disminuye la complejidad algorítmica de $O(n) = N^2$ a solamente $O(n) = n \log^* n$, suponiendo un salto de dos escalones en la Pirámide de Complejidades Comunes:

Notation	Name	Example
$O(1)$	constant	Determining if a binary number is even or odd; Calculating $(-1)^n$; Using a constant-size lookup table
$O(\log \log n)$	double logarithmic	Number of comparisons spent finding an item using interpolation search in a sorted array of uniformly distributed values
$O(\log n)$	logarithmic	Finding an item in a sorted array with a binary search or a balanced search tree as well as all operations in a Bomial heap
$O((\log n)^c)$ $c > 1$	polylogarithmic	Matrix chain ordering can be solved in polylogarithmic time on a Parallel Random Access Machine
$O(n^c)$ $0 < c < 1$	fractional power	Searching in a kd-tree
$O(n)$	linear	Finding an item in an unsorted list or in an unsorted array; adding two n -bit integers by ripple carry
$O(n \log^* n)$	n log-star n	Performing triangulation of a simple polygon using Seidel's algorithm, or the union-find algorithm. Note that $\log^*(n) = \begin{cases} 0, & \text{if } n \leq 1 \\ 1 + \log^*(\log n), & \text{if } n > 1 \end{cases}$
$O(n \log n) = O(\log n!)$	linearithmic, loglinear, or quasilinear	Performing a fast Fourier transform; Fastest possible comparison sort; heapsort and merge sort
$O(n^2)$	quadratic	Multiplying two n -digit numbers by a simple algorithm; simple sorting algorithms, such as bubble sort, selection sort and insertion sort; (worst case) bound algorithms such as quicksort, Shellsort, and tree sort
$O(n^c)$	polynomial or algebraic	Tree-adjoining grammar parsing; maximum matching for bipartite graphs; finding the determinant with LU decomposition
$L_n(\alpha, c) = e^{(c+o(1))(\log n)^{\alpha+1}}$ $0 < \alpha < 1$	L -notation or sub-exponential	Factoring a number using the quadratic sieve or number field sieve
$O(c^n)$ $c > 1$	exponential	Finding the (exact) solution to the traveling salesman problem using dynamic programming; determining if two logical statements are equivalent using truth tables
$O(n!)$	factorial	Solving the traveling salesman problem via brute-force search; generating all unrestricted permutations of a poset; finding the determinant with Laplace's rule

Figura LXVII: Salto de complejidad *cuadrática* a *n log-star n* en la Pirámide de Complejidades Comunes [Fuente: Wikipedia]

De esta forma, se consigue aumentar en mayor medida el rendimiento algorítmico de IFA mediante la reducción de su complejidad en términos de la Notación Bachmann-Landau.

3.2.3 Mejora de la complejidad espacial y datos aleatorios

Tal y como se aprecia en [Yin et al, 2015], el algoritmo necesita llevar a cabo ciertos cálculos en lo referido a su sección de inicialización. (Más en concreto, en la etapa de selección de N ubicaciones iniciales.) Se destaca que el uso de datos aleatorios y pre-inicializados en este punto parece arrojar los mismos resultados de rendimiento sin perder precisión en la solución.

A fin de examinar el rendimiento del algoritmo, este debe ser ejecutado un número de veces suficientemente alto, con distintos *inputs* de información o conjuntos de entrada. Por este motivo, y recordando lo que se ha reseñado varias veces a lo largo del presente epígrafe, es crítico reducir el consumo de recursos en el cuerpo del algoritmo, entendiendo como tal su fase más intensiva en cómputo. A fin de minimizar tal impacto, se destaca que las variables temporales del software han sido reducidas a su mínima expresión, eliminadas o reusadas para evitar dobles instanciaciones cuando ha sido posible. Este proceso de mejora algorítmica ha sido llevado a cabo en el código fuente de manera especial en la etapa de evaluación de la calidad de las ubicaciones de sparks. Una vez más, se referencia al repositorio *GitHub* dedicado para más información.

3.2.4 Diagrama de Flujo

Una vez presentadas con detalle las mejoras llevadas a cabo sobre FA, dando lugar a IFA, es interesante estudiar cómo ha impactado dicho proceso al Diagrama de Flujo algorítmico. Para ello, se recuerda al lector el Diagrama de Flujo de FA:

Figura LXVIII: Diagrama de Flujo de un Fireworks Algorithm (FA)

Así como por otro lado, y a modo de información complementaria, su algoritmo director, detallado al comienzo de este mismo capítulo:

Algorithm 1 – The general structure of FWA.

```
1: Initialize  $N$  fireworks and set the constant parameters
2: repeat
3:   // Explosion operator
4:   Calculate the explosion amplitudes
5:   Calculate the numbers of explosion sparks
6:   Generate “explosion sparks” (check if out-of-bounds)
7:   Evaluate the fitness
8:   // Gaussian mutation operator
9:   Generate “Gaussian sparks” (check if out-of-bounds)
10:  Evaluate the fitness
11:  // Selection strategy
12:  Select fireworks for the next iteration
13: until termination criterion is met.
```

Algoritmo II (2): Estructura general de FA

Como podrá apreciarse a continuación, las mejoras expuestas a lo largo de este epígrafe desembocarán en la creación de dos nuevos estados en el diagrama; *Gaussian Explode For FA Sparks* y *Divide & Conquer*.

Figura LXIX: Diagrama de Flujo de un Improved Fireworks Algorithm (IFA)

3.3 Metodología

En el presente capítulo, basado en explicar qué se ha mejorado y cómo, el epígrafe que nos ocupa consta de un interés especial. Es reseñable, dado que ACORD es un híbrido teórico (se referencia al lector a su apartado dedicado para mayor información), que el presente apartado se centrará en describir cómo es factible comparar FA e IFA, qué mecanismos de *benchmarking* y métricas de *performance* han sido utilizadas en pos de la consecución de tal objetivo y cómo podemos comparar las distintas funciones de *benchmarking* utilizadas, cuyas características varían en un amplio espectro. Adviértase que, dada su naturaleza más puramente numérica, el binomio FA-IFA presenta una carga metodológica mucho mayor en comparación a la de ACORD, siendo la metodología asociada a

este último detallada en su epígrafe dedicado, por lo que se obviará del presente epígrafe para evitar redundancias.

Por otro lado, y de manera especial, se detallarán, dada su importancia, las distintas funciones de *benchmarking* utilizadas para comparar la efectividad de ambos algoritmos. Es reseñable que esta aproximación comparativa entre algoritmos es un enfoque ampliamente aceptado, como demuestra [Mirjalili et al., 2016], que utiliza de igual manera un conjunto de funciones de *benchmarking* para estudiar la diferencia en, términos de performance, de los algoritmos involucrados.

3.3.1 Condiciones y parámetros de la comparativa

Persiguiendo el máximo rigor posible, se pretende comparar FA e IFA tan exhaustivamente como sea computacionalmente factible. Por consiguiente, la comparación ha sido ejecutada en el mismo ordenador y bajo las mismas circunstancias, asegurando la igualdad de condiciones, desactivación de memoria caché, inexistencia de procesos ejecutándose en *background*, etc.

Sin más dilación, este epígrafe resume los parámetros utilizados para ambos algoritmos:

- FA: Técnica de explosión clásica [Yin, 2015], con un número de fireworks de 100 y rango perteneciente al conjunto {100, 100, 30, 32, 600, 5.12, 50, 5, 2, 100, 5.12, 65.536}. Las dimensiones disponibles, en las que el algoritmo puede operar vienen dadas por el vector discreto {30, 30, 30, 30, 30, 30, 30, 2, 2, 2, 30, 30}.
- IFA: Tal y como se detalla pormenorizada y exhaustivamente en el presente capítulo, el algoritmo aplica la variación de la técnica de explosión Gaussiana. Al igual que su competidor, el número de fireworks es de 100. El conjunto de rangos, así como las dimensiones disponibles para que el algoritmo opere, son exactamente las mismas que en FA para asegurar la igualdad de condiciones en la comparativa.

Es importante destacar, de cara a la comparativa, que se medirá el tiempo que tarda cada algoritmo (FA e IFA) en encontrar el punto óptimo para cada función de *benchmarking*.

3.3.2 Benchmarking y Métricas de Rendimiento

El conjunto de funciones de benchmarking aplicado a ambos algoritmos ha sido cuidadosamente seleccionado a partir de métodos ampliamente aceptados para la prueba de optimizaciones [Oldenhuis, 2009] [Bäck, 1995].

Es especialmente reseñable que en [Xue et al., 2016] se compara de igual manera una variación algorítmica realizada sobre FA, sentando precedente en la comparación empírica de algoritmos mediante funciones clásicas de benchmarking.

Llegados a este punto, se presentarán en los subsiguientes epígrafes los aspectos fundamentales y/o más interesantes de dichas funciones, obviando aquellas que sean simples variaciones de las aquí descritas.

3.3.2.1 Función Ackley

Función de forma bidimensional, caracterizada por un área externa idealmente llana que presenta un punto mínimo en el centro [Adorio et al., 2005]. Es reseñable que esta función de benchmark es especialmente peligrosa en los aspectos referidos a algoritmos de optimización puros, más concretamente del tipo *hill climbing*, puesto que tienden a estancarse en uno de los numerosos mínimos locales. La función Ackley, típicamente evaluada para $X_i \in [-32.768, 32.768]$, para todo $i \in [1, 2, \dots, d]$, viene dada por la fórmula:

$$f(\mathbf{x}) = -a \exp \left(-b \sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2} \right) - \exp \left(\frac{1}{d} \sum_{i=1}^d \cos(cx_i) \right) + a + \exp(1)$$

En este caso particular, el mínimo global puede encontrarse, de manera trivial, en el punto:

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0, \dots, 0)$$

La función Ackley se representa en la siguiente figura, donde las características anteriormente expuestas pueden apreciarse:

Figura LXX: Función Ackley - Representación gráfica bidimensional

3.3.2.2 Función Schwefel

Función que consta de un amplio número de mínimos locales [Laguna et al., 2002], tal y como ocurre con su homóloga Sphere, a su vez. Se evalúa en el hipercubo $X_i \in [-500, 500]$, para todo $i \in [1, 2, \dots, d]$. La función Schwefel viene dada por la fórmula:

$$f(\mathbf{x}) = 418.9829d - \sum_{i=1}^d x_i \sin(\sqrt{|x_i|})$$

La representación de la Función Schwefel, que presenta un mínimo en la siguiente función $f(x)$, puede apreciarse en la siguiente figura:

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (420.9687, \dots, 420.9687)$$

Figura LXXI: Función Schwefel - Representación gráfica bidimensional

3.3.2.3 Función Sphere

La presente función consta de d mínimos locales y uno global [Dixon et al., 1978]. La función a satisfacer viene dada por:

$$f(\mathbf{x}) = \sum_{i=1}^d x_i^2$$

Donde el mínimo global puede ser encontrado en:

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0, \dots, 0)$$

La función Sphere es continua, convexa y unimodal. A continuación puede apreciarse, en la siguiente figura, su forma bidimensional:

Figura LXXII: Función Sphere - Representación gráfica bidimensional

3.3.2.4 Función Rosenbrock

Función unimodal y con forma de valle [Dixon et al., 1978], donde el mínimo global reside en el punto más bajo de la representación gráfica. Tal y como se establece en [Picheny et al., 2012], la convergencia hacia el punto mínimo del valle no es trivial. Rosenbrock, evaluada para el hipercubo $x_i \in [-5, 10]$, para todo $i \in [1, 2, \dots, d]$, viene dada por la siguiente fórmula:

$$f(\mathbf{x}) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

Es reseñable que Rosenbrock también presenta un mínimo global dado por la siguiente función:

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (1, \dots, 1)$$

Figura LXXIII: Función Rosenbrock - Representación gráfica bidimensional

3.3.2.5 Función Griewanks

Función de benchmarking que comparte su mínimo global con su homóloga Ackley. La función Griewanks [Molga et al., 2005], que presenta puntos mínimos regularmente situados, se evalúa, normalmente, para $X_i \in [-600, 600]$, para todo $i \in [1, 2, \dots, d]$, y viene dada por la fórmula:

$$f(\mathbf{x}) = \sum_{i=1}^d \frac{x_i^2}{4000} - \prod_{i=1}^d \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

La representación gráfica de la función Griewanks puede estudiarse en la siguiente figura:

Figura LXXIV: Función Griewanks - Representación gráfica bidimensional

3.3.2.6 Función Rastrigin

Esta función, que presenta un elevado número de mínimos locales, se comporta de acuerdo a la fórmula [Bäck, 1995]:

$$f(\mathbf{x}) = 10d + \sum_{i=1}^d [x_i^2 - 10 \cos(2\pi x_i)]$$

La función Rastrigin tiende a ser evaluada para $x_i \in [-5.12, 5.12]$, para todo $i \in [1, 2, \dots, d]$ y comparte su mínimo local con Ackley y Griewanks. La representación gráfica de la función Rastrigin puede apreciarse en la siguiente figura:

Figura LXXV: Función Rastrigin - Representación gráfica bidimensional

3.3.2.7 Función Goldstein

La función Goldstein, normalmente evaluada para $X_i \in [-2, 2]$, para todo $i \in [1, 2]$, sigue la fórmula [Dixon et al., 1978]:

$$f(\mathbf{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

Por consiguiente, su mínimo global puede encontrarse en:

$$f(\mathbf{x}^*) = 3, \text{ at } \mathbf{x}^* = (0, -1)$$

La función Goldstein puede ser gráficamente representada tal y como se aprecia en la siguiente figura. No obstante, es reseñable que existen numerosas variaciones [Plcheny, 2012] en lo referido a la representación de la función Goldstein, como por ejemplo la Forma Goldstein Re-escalada con media de cero y varianza uno.

Figura LXXVI: Función Goldstein - Representación gráfica bidimensional

3.3.2.8 Función Schaffer

La presente función [Simonescu et al., 2002], evaluada para $X_i \in [-100, 100]$, para todo $i \in [1, 2]$, respeta la fórmula:

$$f(\mathbf{x}) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$$

En lo referido al mínimo global de la función Schaffer, y tal y como se ha descrito previamente en numerosas funciones de benchmarking, este puede encontrarse en:

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0, 0)$$

Finalmente, es reseñable que la representación gráfica de Schaffer se presenta en la siguiente figura:

Figura LXXVII: Función Schaffer - Representación gráfica bidimensional

3.3.2.9 Función Rotated Hyper Ellipsoid (Sum Squares)

Esta función continúa, multimodal y convexa [Molga et al., 2005], que puede verse como una variación de la función Hyper Ellipsoid [Dixon et al., 1978], se evalúa para $X_i \in [-65.536, 65.536]$, para todo $i \in [1, 2, \dots, d]$. La presente función respeta la siguiente fórmula, y comparte el mínimo Global con Schaffer en $f(\mathbf{x}) = 0$:

$$f(\mathbf{x}) = \sum_{i=1}^d \sum_{j=1}^i x_j^2$$

Por otro lado, en lo referido a su representación gráfica, la función Rotated Hyper Ellipsoid consta de la forma que se aprecia en la siguiente figura:

Figura LXXVIII: Función Rotated Hyper Ellipsoid - Representación gráfica bidimensional

3.3.3 Comparativa entre Funciones de Benchmarking

Es reseñable que las funciones de benchmarking pormenorizadamente descritas en el anterior epígrafe, presentan un rango de evaluación tremendamente dispar. Por citar alguna, mientras que la función Goldstein puede ser evaluada en el rango $[-2, 2]$, Rotated Hyper Ellipsoid (Sum Squares) puede hacer lo propio en el intervalo $[-65.536, 65.536]$. Esta casuística, presente a lo largo de todo el grupo de funciones de benchmarking, asegura un conjunto de datos enriquecido y dispar para la extracción de resultados concluyentes.

Asimismo, la complejidad de las distintas funciones de benchmarking varía en un amplio espectro, permitiendo a los algoritmos ser probados en funciones matemáticamente fáciles de computar (como la función Shaffer, por ejemplo), y en homólogas complejas, que exigen el máximo de la unidad aritmético-lógica de la CPU (como es Penalized Six-Hump Camel Back). En lo referido a la evaluación de la información arrojada, el rendimiento del algoritmo (FA e IFA) será inversamente proporcional al tiempo utilizado para la ejecución de cada función de benchmarking. Por consiguiente, y en otras palabras, los valores temporales bajos indicarán un mejor rendimiento.

4. RESULTADOS

A lo largo del presente capítulo se presentan los resultados arrojados por el experimento en cuestión: la ejecución de todas las funciones de *benchmarking* (y variaciones de las mismas) anteriormente expuestas para FA e IFA, con la correspondiente medición temporal para cada *shift* o tanda. Se recuerda, de cara a la comparativa, que se medirá el tiempo que tarda cada algoritmo (FA e IFA) en encontrar el punto óptimo para cada función de *benchmarking*, siendo estos los tiempos que aparecen en el epígrafe actual.

Por otro lado, cabe destacar que el análisis de tales resultados se llevará a cabo en el siguiente capítulo, restringiendo este al más puro muestreo y presentación de resultados. Es reseñable que, debido a la índole teórica del Algoritmo ACORD, y a fin de evitar caer en redundancias, se referencia al epígrafe 3.1.2 *Resultados con Santa Fe Trail* para el estudio de los resultados del mismo, dedicando esta sección exclusivamente a FA e IFA, cuyos resultados numéricos son más abundantes y susceptibles al estudio que los de ACORD.

Retomando la misión del presente capítulo, se destaca que la información normalizada para cada función de benchmarking se presenta, para cada algoritmo, en las siguientes tablas. Dicho resultado se divide en dos partes bien diferenciadas: una para FA y otra para su competidor, IFA. Para cada una, cada entrada representa el tiempo que cada algoritmo ha necesitado para realizar la función de benchmarking $f(x)$. Este proceso se repite siete veces por función en el ejemplo, a fin de mantener el sesgo al nivel mínimo. Téngase a su vez en cuenta que los resultados aquí presentes se asocian con una ejecución de entre un amplio elenco, cuando el algoritmo arroja resultados estables constantemente, a fin de garantizar la igualdad de condiciones para FA e IFA. Más en concreto, los resultados de cada tabla se corresponden con datos extraídos a partir de 1K ejecuciones, con el algoritmo más estabilizado. Al final de cada tabla, puede consultarse el tiempo total medio consumido por cada algoritmo, FA e IFA, para 10K ejecuciones, siendo especialmente importante esta medida para averiguar con efectividad la variación de rendimiento entre los algoritmos.

Es especialmente reseñable que, a tenor del *No Free Lunch Theorem for Optimization* [Wolpert et al., 1997] no existe una “bala de plata” en lo referido a un hipotético mejor algoritmo de optimización. En otras palabras [Caruana et al., 2006], incluso los mejores modelos (en Teoría de Grafos, especialmente árboles calibrados y bosques aleatorios con distintas características en dicho estudio) presentan un pobre rendimiento en algunos problemas, mientras que modelos que constan de un rendimiento paupérrimo en términos de media aritmética, disfrutan de un alto rendimiento en ciertas casuísticas, problemas, escenarios o

métricas. Esta teoría se retomará en el siguiente capítulo, dedicado al análisis de los resultados en su totalidad.

Finalmente, antes de presentar el rendimiento asociado a cada función de benchmarking, es importante destacar que se ha ejecutado de forma previa una suerte de *fase de calentamiento hardware*, a fin de velar por unos resultados precisos y ecuanímenes para ambos algoritmos. Además, tanto FA como IFA han sido ejecutados bajo absolutamente idénticas condiciones a fin de asegurar esta precisión en los resultados. Sin más dilación, se presentan las tablas normalizadas para cada función de benchmarking, que contienen el tiempo que tarda cada algoritmo en encontrar el valor óptimo de cada función $f(x)$ en varias ejecuciones (*shifts*), medido en segundos (s):

Benchf(x)	Shift 0(s)	Shift 1(s)	Shift 2(s)	Shift 3(s)	Shift 4(s)	Shift 5(s)	Shift 6(s)
Sphere	7.9313E-5	1.0574E-4	1.1105E-4	8.0312E-5	1.1238E-4	1.7834E-5	8.1238E-4
Schwefel	0.0014	0.0016	0.0014	0.0013	0.0015	0.0016	0.0016
Rosenbr.	0.2711	0.3469	0.5677	0.2712	0.5955	0.3946	0.2929
Ackley	2.0352E-8	2.0921E-8	2.1005E-8	2.0953E-8	2.1983E-8	2.0949E-8	2.1986E-8
Griew.	8.0606E-7	1.0875E-6	8.6277E-7	8.0734E-7	8.6425E-7	1.0903E-6	8.4595E-7
Rastrigin	4.0081E-7	3.7058E-7	4.2390E-7	4.3039E-7	4.2533E-7	3.8729E-7	4.0291E-7
Penalized	1.6874	2.6238	3.2342	1.2345	3.9234	2.7823	1.7830
Six Hump	2.3440E-7	4.3273E-7	7.7866E-7	2.3638E-7	7.7939E-7	5.3923E-7	3.5923E-7
Goldstein	6.2323E-7	5.1948E-7	4.8160E-7	6.2522E-7	4.8313E-7	5.4292E-7	7.3494E-7
Schaffer	4.9189E-10	4.9331E-10	4.9462E-10	4.9302E-10	4.9734E-10	4.9829E-10	4.9192E-10
Axis Par.	1.5512E-6	1.8168E-6	2.2459E-6	1.5549E-6	2.1492E-6	1.92301E-6	1.7902E-6
Rot H. E.	0.0050	0.0055	0.0067	0.0051	0.0069	0.0056	0.0059
TOTAL(s)	1,97	2,98	3,81	1,51	4,50	3,18	2,14

Average execution time/Shift (10K, s)	2,94891254814 = 2,95 s
Total execution time (10K, h)	8,1914 h

Tabla V: Tiempo de ejecución (s) para cada par Benchmark function-Shift y tiempo total necesitado para FA

Benchf(x)	Shift 0(s)	Shift 1(s)	Shift 2(s)	Shift 3(s)	Shift 4(s)	Shift 5(s)	Shift 6(s)
Sphere	6.2538E-5	1.0626E-4	1.0196E-4	1.3209E-4	1.0821E-4	6.4738E-5	8.9034E-5
Schwefel	0.0011	0.0012	0.0015	0.0016	0.0014	0.0017	0.0011
Rosenbr.	0.2782	0.3386	0.5319	0.4329	0.3983	0.2839	0.3982
Ackley	2.0433E-8	2.0761E-8	2.0398E-8	2.0943E-8	2.0832E-8	2.0250E-8	2.0423E-8
Griew.	7.4914E-7	8.6181E-7	8.0224E-7	8.2339E-7	7.9990E-7	8.7590E-7	8.0591E-7
Rastrigin	3.6800E-7	3.8044E-7	3.7079E-7	3.7338E-7	3.8221E-7	3.7539E-7	3.8420E-7
Penalized	1.5049	2.2231	3.4099	2.9492	2.9191	3.4019	1.4912
Six Hump	2.0298E-7	4.0204E-7	7.0342E-7	7.3392E-7	4.3032E-7	2.8310E-7	2.3028E-7
Goldstein	4.9083E-7	3.8014E-7	4.0882E-7	4.5439E-7	3.8993E-7	4.4120E-7	3.9292E-7
Schaffer	4.9228E-10	4.9369E-10	4.9528E-10	4.9932E-10	4.9923E-10	4.9903E-10	4.9302E-10
Axis Par.	1.1960E-6	1.7734E-6	1.9919E-6	1.6453E-6	1.8349E-6	1.2041E-6	1.3024E-6
Rot H. E.	0.0052	0.0046	0.0063	0.0069	0.0048	0.0060	0.0043
TOTAL(s)	1,79	2,56	3,90	3,39	3,32	3,69	1,89

Average execution time/Shift (10K, s)	2.064238783 = 2,06 s
Total execution time (10K, h)	5,7222 h

Tabla VI: Tiempo de ejecución (s) para cada par Benchmark function-Shift y tiempo total necesitado para IFA

Finalmente en lo referente al presente capítulo, y a modo de resumen, se presenta un ilustrativo extracto de los resultados en la siguiente tabla, con decimales aproximados. Dicha tabla, que aúna la primera terna de *shifts*, para FA e IFA, muestra a su vez los tiempos de ejecución totales (tiempo necesitado para encontrar el punto óptimo en cada función de benchmark) pertinentes para cada uno, con el mejor tiempo por *shift* (es decir, el menor tiempo) resaltado. De esta forma, se establece un prólogo para el siguiente capítulo, dedicado en exclusiva al análisis de los resultados mostrados en su homólogo actual:

	FA			IFA		
Bench. $f(x)$	Shift 0 (s)	Shift 1 (s)	Shift 2 (s)	Shift 0 (s)	Shift 1 (s)	Shift 2 (s)
Sphere	7.9313E-5	<u>1.0574E-4</u>	1.1105E-4	<u>6.2538E-5</u>	1.0626E-4	<u>1.0196E-4</u>
Schw.	0.0014	0.0016	<u>0.0014</u>	<u>0.0011</u>	<u>0.0012</u>	0.0015
Rosenbr.	<u>0.2711</u>	0.3469	0.5677	0.2782	<u>0.3386</u>	<u>0.5319</u>
Ackley	<u>2.0352E-8</u>	2.0921E-8	2.1005E-8	2.0433E-8	<u>2.0761E-8</u>	<u>2.0398E-8</u>
Griew.	8.0606E-7	1.0875E-6	8.6277E-7	<u>7.4914E-7</u>	<u>8.6181E-7</u>	<u>8.0224E-7</u>
Rastrig.	4.0081E-7	<u>3.7058E-7</u>	4.2390E-7	<u>3.6800E-7</u>	3.8044E-7	<u>3.7079E-7</u>
Penaliz.	1.6874	2.6238	<u>3.2342</u>	<u>1.5049</u>	<u>2.2231</u>	3.4099
Six H.	2.3440E-7	4.3273E-7	7.7866E-7	<u>2.0298E-7</u>	<u>4.0204E-7</u>	<u>7.0342E-7</u>
Goldst.	6.2323E-7	5.1948E-7	4.8160E-7	<u>4.9083E-7</u>	<u>3.8014E-7</u>	<u>4.0882E-7</u>
Schaff.	<u>4.9189E-10</u>	<u>4.9331E-10</u>	<u>4.9462E-10</u>	4.9228E-10	4.9369E-10	4.9528E-10
Axis Par.	1.5512E-6	1.8168E-6	2.2459E-6	<u>1.1960E-6</u>	<u>1.7734E-6</u>	<u>1.9919E-6</u>
Rot. H. E.	<u>0.0050</u>	0.0055	0.0067	0.0052	<u>0.0046</u>	<u>0.0063</u>

Tabla VII: Tiempo total de ejecución (s) para cada par Benchmark function-Shift y tiempo total necesitado para FA e IFA

5. ANÁLISIS DE LOS RESULTADOS Y DISCUSIÓN

El objetivo atómico del presente capítulo es el de analizar los resultados mostrados en su homólogo anterior. La cantidad de tiempo necesitada por FA, así como la propia para su versión mejorada, IFA, en ejecutar las funciones de benchmarking descritas (en otras palabras, el tiempo que tarda cada algoritmo en encontrar el punto óptimo de cada función de benchmark), así como la comparación entre los resultados para ambos algoritmos en términos porcentuales, será minuciosamente relatada.

Los resultados empíricos son sorprendentes. A modo de avance: más de un 40% de efectividad en la versión mejorada, fruto de una renovada concepción en cuanto al comportamiento y modo de funcionamiento interno de FA.

A fin de comparar de manera empírica el rendimiento de FA frente a IFA, se recurrirá a la Ley de Amdahl [Amdahl, 1967]. Dicha ley mide la mejora, en términos porcentuales, de un algoritmo sobre otro como:

$$S_{\text{latency}}(s) = \frac{1}{(1 - p) + \frac{p}{s}}$$

Donde S_{latency} se corresponde con la aceleración teórica de la ejecución de una tarea objeto de estudio, s es la aceleración de la sección de la tarea que se beneficia de las posibles mejoras y p es la proporción del tiempo de ejecución que la parte beneficiada de los recursos mejorados ocupaba de manera original. Por consiguiente, es lícito inferir que:

$$\begin{cases} S_{\text{latency}}(s) \leq \frac{1}{1 - p} \\ \lim_{s \rightarrow \infty} S_{\text{latency}}(s) = \frac{1}{1 - p} \end{cases}$$

Dicha función demuestra que la aceleración teórica del tiempo de ejecución de la tarea completa incrementa con la mejora de los recursos del sistema, y que de forma independiente a la magnitud de la mejora, la aceleración teórica siempre se ve limitada por la parte de la tarea que no puede beneficiarse de la mejora [Rodgers, 1985].

Una consecuencia del anterior razonamiento es, lógicamente, que una tarea ejecutada por un sistema cuyos recursos se mejoran puede ser escindida en dos partes:

1. La parte que no se beneficia de la mejora de los recursos del sistema.
2. La contrapuesta, que sí lo hace.

Un ilustrativo ejemplo del escenario en cuestión puede ser el de un software que procesa archivos desde disco. Una parte de dicho software escaneará el directorio del disco para crear una lista de archivos en memoria interna. Después, otra parte del software pasará cada archivo a un thread de procesamiento independiente. En este ejemplo, mientras que la parte que escanea el directorio y crea los archivos en memoria no puede acelerarse en un sistema monolítico, su homóloga de procesamiento de ficheros en threads independientes de ejecución puede.

La ejecución de la tarea completa antes de la mejora de los recursos del sistema puede, por tanto, denotarse T . Es reseñable que incluye el tiempo de ejecución de la parte que no se beneficiará de la mejora, y el tiempo de ejecución de la que sí se beneficiará. La fracción del tiempo de ejecución de la tarea que se verá beneficiada por la mejora en los recursos se denota por p , implícitamente denotando a su homóloga, que no se verá beneficiada, por el complementario en términos estadísticos, $1-p$. Entonces:

$$T = (1 - p)T + pT.$$

Es la ejecución de la parte beneficiada la que se ve acelerada por un factor s después de la mejora, por lo que el tiempo de ejecución de la parte no beneficiada permanece inmutable, mientras que la parte beneficiada pasa a ser:

$$\frac{p}{s}T.$$

Recapitulando, el tiempo de ejecución teórico $T(s)$ de la tarea completa después de la mejora de los recursos del sistema es:

$$T(s) = (1 - p)T + \frac{p}{s}T.$$

Obteniendo, gracias a la Ley de Amdahl, la aceleración teórica en la latencia de la ejecución de la tarea completa con una carga de trabajo idéntica, W , lo que se traduce en:

$$S_{\text{latency}}(s) = \frac{TW}{T(s)W} = \frac{T}{T(s)} = \frac{1}{1 - p + \frac{p}{s}}.$$

Fórmula de la cual puede inferirse la expresión que mide el porcentaje de mejora en términos porcentuales:

$$\text{percentage improvement} = 100 \left(1 - \frac{1}{S_{\text{latency}}} \right).$$

Trivialmente, esta expresión puede reducirse a la siguiente fórmula, que permitirá aplicar los resultados obtenidos:

$$\text{Percentage Improvement} = \frac{\text{Non - Improved Time}}{\text{Improved Time}}$$

Por consiguiente, sustituyendo los valores arrojados por la medición de tiempo en la ejecución de FA e IFA (Capítulo 4. *Resultados*), se obtiene (advírtase que se utilizan, en este caso, los tiempos en nanosegundos, con todos sus decimales, para conseguir la mayor precisión posible):

$$\text{Percentage Improvement} = \frac{2948912548,14}{2064238783,7} = 1,428571429$$

Por consiguiente, y a tenor de la Ley de Amdahl, es lícito inferir que IFA mejora a FA en un 42.8571429%. A modo de conclusión, la tabla definitiva, que contiene los resultados de sustraer a FA el valor de IFA para cada par tiempo de ejecución (s) Benchmark function/*Shift* de las tablas anteriores, se presenta seguidamente. (En otras palabras, para cada $f(x)$, por ejemplo *Sphere*, se sustrae al tiempo de *Sphere* (FA) el valor de *Sphere* (IFA)). En ella, se muestra el valor de mejora para cada evaluación, pudiendo dicho valor circunscribirse a uno de los siguientes escenarios:

1. Valor positivo (resaltado en verde): Representa el tiempo que IFA fue más rápido (mayor rendimiento) que FA.
2. Valor negativo (resaltado en amarillo): Caso opuesto. (FA rinde mejor que IFA.)

Benchf(x)	Shift 0(s)	Shift 1(s)	Shift 2(s)	Shift 3(s)	Shift 4(s)	Shift 5(s)	Shift 6(s)
Sphere	1.68E-05	-5.20E-07	9.09E-06	-5.18E-05	4.17E-06	-4.69E-05	7.23E-04
Schwefel	0.0003	0.0004	-0.0001	-0.0003	0.0001	-0.0001	0.0005
Rosenbr.	-7.10E-03	8.30E-03	3.58E-02	-1.62E-01	1.97E-01	1.11E-01	-1.05E-01
Ackley	-8.10E-11	1.60E-10	6.07E-10	1.00E-11	1.15E-09	6.99E-10	1.56E-09
Griewanks	5.69E-08	2.26E-07	6.05E-08	-1.60E-08	6.44E-08	2.14E-07	4.00E-08
Rastrigin	3.28E-08	-9.86E-09	5.31E-08	5.70E-08	4.31E-08	1.19E-08	1.87E-08
Penalize.	1.83E-01	4.01E-01	-1.76E-01	-1.71E+00	1.00E+00	-6.20E-01	2.92E-01
Six Hump	3.14E-08	3.07E-08	7.52E-08	-4.98E-07	3.49E-07	2.56E-07	1.29E-07
Goldstein	1.32E-07	1.39E-07	7.28E-08	1.71E-07	9.32E-08	1.02E-07	3.42E-07
Schaffer	-3.90E-13	-3.80E-13	-6.60E-13	-6.30E-12	-1.89E-12	-7.40E-13	-1.10E-12
Axis Par.	3.55E-07	4.34E-08	2.54E-07	-9.04E-08	3.14E-07	7.19E-07	4.88E-07
Rot.H.E.	-0.0002	0.0009	0.0004	-1.80E-03	2.10E-03	-4.00E-04	5.47E-02

Total execution time (%)	42.8571429
--------------------------	-------------------

Tabla VIII: Tiempo de mejora (s) para cada par Benchmark function-Shift al restar FA-IFA y porcentaje total de mejora entre FA e IFA por la Ley de Amdahl

A modo de conclusión, la tabla muestra que, a pesar de que FA sigue presentando un mejor rendimiento en ciertos Shifts o Turnos, cuando IFA presenta más rendimiento, lo hace con una gran diferencia y en más casos. Tómese por ejemplo de la aseveración anterior la función Axis Parallel, por citar una, donde FA únicamente presenta un mayor rendimiento que IFA en el Shift 3, aproximadamente 90 ns de diferencia, mientras que en el resto de casos IFA supera a FA (en el Shift 5, por 718 ns de diferencia, valor más de 10 veces mayor que la ganancia contraria del Shift 4). Los Shifts restantes, sin embargo, otorgan una clara victoria a IFA en su mayoría, en ciertos casos con una diferencia mastodóntica, como puede apreciarse en el Shift 5, como se indicaba anteriormente (718.9735412 ns).

Es reseñable recordar que el anterior razonamiento vuelve a cargar de razón al *No Free Lunch Theorem for Optimization* [Wolpert et al., 1995], ya descrito pormenorizadamente en epígrafes anteriores, el cual demuestra que la versión menos optimizada de un algoritmo puede presentar un mayor rendimiento en ciertas casuísticas puntuales del problema a resolver. La función Schaffer es un claro ejemplo de la anterior aseveración, puesto que, a pesar de que IFA presenta un rendimiento mayor (con diferencia) en 58 de los 84 (69.04%) casos totales que presenta la tabla, no presenta un rendimiento mayor en cuanto a la función Schaffer. Es reseñable, no obstante, que la diferencia es tan microscópica que FA e IFA bien podrían considerarse empatadas en cuanto a rendimiento en lo referido a la función Schaffer.

Dado que la tabla resaltada ofrece, de manera presta y panorámica, una clara visión del desempeño presentado por ambos algoritmos, se evitará caer en redundancias de análisis, destacando, especialmente, los siguientes razonamientos, que condensan la tabla:

- En términos cuantitativos, IFA presenta un mayor rendimiento en 58 de los 84 casos (69.04%).
- En la misma línea, y de manera complementaria, FA presenta un mayor rendimiento en 26 de los 84 casos (30,96%).
- A nivel cualitativo, la diferencia entre IFA y FA es abismal; si FA es mejor que IFA, es por una diferencia despreciable, que podría considerarse nula (*Schaffer*), mientras que cuando IFA es mejor, es por una diferencia abrumadora (como demuestran especialmente las funciones *Griewanks*, *Axis-Parallel*, *Goldstein*, *Rastrigin* o *Ackley*.)

Es reseñable que FA disfruta, en términos medios, de un rendimiento superior a metaheurísticas como PSO en numerosos escenarios, por lo que IFA, estando por encima de ambos y a tenor de los resultados analizados en el presente epígrafe, ocuparía un privilegiado lugar en el tercio superior de la siguiente tabla, que muestra una comparativa entre numerosos algoritmos evolutivos de optimización [Elbeltagi et al., 2005]:

Tabla IX: Tabla comparativa entre *running length* y *proportion of functions* de varios algoritmos evolutivos de optimización [Elbeltagi et al., 2005]

Una vez demostrados los resultados arrojados por el experimento, y habiendo ilustrado en más de un 42% la mejora de IFA sobre FA, es materia del siguiente epígrafe el análisis de las líneas futuras de investigación y su aplicación a smart cities, donde no sólo se tratarán ampliaciones con el ecosistema de *Fireworks Algorithm* como protagonista, sino de todo el amplio espectro asociado al Paradigma de la Computación Natural.

6. LÍNEAS FUTURAS DE INVESTIGACIÓN

En enero de 2018, el prestigioso MIT estadounidense incluyó la técnica *Gene Therapy 2.0* [Mullin, 2018], que se apoya en el paradigma de la Computación Natural, en su lista de las 10 tecnologías emergentes que cambiarán el mundo. De manera subyacente a esta decisión, y entre otros factores, se encuentra la increíble amplitud del elenco de aplicaciones del que disfruta este paradigma, razón por la que las líneas futuras de investigación en las que se puede aplicar el mismo están, casi únicamente, acotadas por la imaginación y la creatividad.

Llegado el momento de pensar en el futuro, es obligado deliberar qué necesidades tendrán la comunidad científica y la sociedad, así como las líneas de investigación que colaborarán en la satisfacción de dichas necesidades. Por esta razón, se relatan en el presente epígrafe las posibles líneas futuras de investigación teóricas fruto de la Tesis Doctoral que nos ocupa.

Tal y como puede intuirse, el elenco de líneas futuras de investigación y desarrollo de la presente Tesis Doctoral es inmenso, razón por la que el análisis de dichas líneas se centrará en la teoría.

El amplio número de entidades adscritas al Paradigma de la Computación Natural, especialmente propensas a las sinergias dadas sus características, constituye un fructífero campo de investigación, en el que destacan [Fernandez-Marquez et al., 2013], [Chen et al., 2014], [Dam, 2013], [Patitz, 2014], [Gheorghe et al., 2012], [Babaei, 2013], todos ellos centrados en la aplicación de entidades algorítmicas y/o técnicas adscritas a la Computación Natural para la resolución de problemas de optimización.

A lo largo de este epígrafe, se plantearán varios híbridos entre algoritmos naturales, así como una posible ampliación de la propia Tesis Doctoral a acometer en el futuro.

6.1 Esquema Híbrido entre FA & GA

Estableciendo un símil entre el proceso evolutivo de una población de genotipos hacia un fenotipo o solución óptima, los Algoritmos Genéticos (*Genetic Algorithms*; GA) han sido pormenorizadamente desmenuzados en la presente Tesis Doctoral.

Es destacable que la similitud entre GA y FA viene definida por los operadores de mutación (*mutation*) y selección (*crossover*). A pesar de que cada uno actúa de forma sucintamente similar, a excepción de, por ejemplo, la etapa de

selección, donde FA selecciona soluciones basadas en las distancias entre las mismas mientras que GA utiliza la denominada *fitness function*, la principal diferencia entre ambos algoritmos reside en cómo generan las nuevas soluciones.

En GA, la solución se corresponde con un conjunto cromosómico, mientras que, en FA, la solución viene en forma de *fireworks*, que lanzarán *sparks* con más o menos éxito.

Es reseñable a su vez que, en FA, el *firework* podría apreciarse, en cierto modo, como la población de soluciones en las que se aplicarán los operadores, mientras que las *sparks* se corresponden con las nuevas soluciones generadas tras aplicar los operadores.

Podría inferirse que, de hecho, GA y FA definen nuevos subconjuntos dentro del ámbito de la Computación Natural: GA es una técnica de algoritmos evolutivos principalmente basada en teoría de juegos dada su naturaleza atómica, puramente competitiva, mientras que FA está basado, en gran parte, en algoritmos de *Swarm Intelligence*, factor que desemboca en un gran desempeño en problemas de optimización combinatoria dada su naturaleza cooperativa. Adviértase, en este punto, que la diferencia es sutil, pero crucial, puesto que GA se basa en la competición y FA en la cooperación.

Por tanto, y dado que FA está íntimamente ligado al paradigma *Swarm Intelligence*, una interesante línea futura de investigación teórica podría ser la de realizar un proceso de *mapping* entre GA y FA basado en los estudios previos de hibridación entre PSO-GA, entre los que destaca [Garg, 2015]. La siguiente figura ilustra el proceso de mapping, para dos partículas en un determinado instante, en PSO:

Figura LXXIX: Modelo estándar de PSO; actualización posicional utilizando una topología en estrella ($gBest$) y un término inercial, w

Así, se estaría en disposición de, por ejemplo, valorar qué híbrido presenta un mayor rendimiento, proceso en el cual podría utilizarse el conjunto de *benchmarks* presentado en la presente Tesis Doctoral.

Precisamente, y en la línea de los conjuntos de *benchmarking* posibles, destaca la existencia del desenfadado y amplísimo conjunto *Evolutionary Computation Bestiary*, disponible en <http://fcampelo.github.io/EC-Bestiary/>. Tal y como se establece en el mismo, el campo de la búsqueda meta-heurística tiene un largo historial de búsqueda de inspiración en sistemas naturales, desde pájaros y abejas hasta aspectos tan dispares como, precisamente, la explosión de los fuegos artificiales, pasando por campos adscritos al reino de la ficción como los *zombies*. Tal y como se especifica en el propio repositorio:

“The goal of the Evolutionary Computation Bestiary is to catalog the, ermm... exuberance of the meta-heuristic “eco-system”. We try to keep a list of the many different animals, plants, microbes, natural phenomena and supernatural activities that can be spotted in the wild lands of the metaphor-based computation literature.”

En dicho repositorio, prácticamente cualquier método basado en metáforas como las anteriormente citadas está disponible, tanto a nivel referencial como de *benchmarking*. A modo ilustrativo, se destacan las siguientes:

- *A New Meta-Heuristic Optimization Algorithm Inspired by FIFA World Cup Competitions: Theory and Its Application in PID Designing for AVR System* [Razmjoo et al., 2016]: Ofrece una interesante técnica para optimización

de funciones matemáticas basada en la Copa del Mundo de la FIFA, dando lugar al *World Cup Algorithm* (WCU).

- *Grey Wolf Optimizer* [Seyedali et al., 2014]: Los lobos grises (*Canis Lupus*) suponen la inspiración principal del presente trabajo de investigación, que da lugar a una metaheurística homónima, GWO, basada en el espíritu de liderazgo de los lobos grises en la naturaleza.
- *Zombie Survival Optimization: A swarm intelligence algorithm inspired by zombie foraging* [Hoang et al., 2012]: Este original enfoque de enjambre, basado en el comportamiento errático de los *zombies*, propone que estos seres se comporten como agentes exploradores. Tales agentes realizan una búsqueda en un espacio donde la función de *fitness* se modela como un antídoto aéreo que cura a los zombies, transformándolos en humanos de nuevo. (Su implementación, en C++, es pública.)
- *Lightning Search Algorithm* [Shareef et al., 2015]: Nuevo enfoque de optimización metaheurística (LSA) basado en el *rayo*, así como su mecanismo natural de propagación en *steps* o pasos, utilizando el concepto de partículas de propagación rápida denominadas *proyectiles*, mostrando así una eminente reminiscencia con FA, objeto de estudio de la presente Tesis Doctoral.
- *Grasshopper Optimisation Algorithm: Theory and Application* [Saremi et al., 2017]: Los saltamontes adquieren el papel protagonista en GOA, donde el algoritmo modela, a nivel matemático, y mimetiza el comportamiento de los enjambres de saltamontes en la naturaleza en pos de la resolución de problemas de optimización.

Adviértase que el repositorio *Evolutionary Computation Bestiary* puede ser utilizado por prácticamente cualquiera de las líneas de ampliación y mejora presentadas en el presente capítulo, especialmente para aquellas basadas en esquemas híbridos, como la actual.

6.2 Esquema Híbrido entre FA & ACO

Tal y como se explica en numerosas ocasiones a lo largo de esta Tesis Doctoral, un factor común en el ámbito de la algorítmica bioinspirada y la Computación Natural es la tendencia a la hibridación y la sinergia entre las entidades que la forman.

En tal sentido, Fireworks Algorithm encuentra un gran aliado en Ant Colony Optimization, que bien podría ser una línea de investigación teórica (con cierta

vertiente práctica, como se verá en este epígrafe) fruto de la presente Tesis Doctoral. Supóngase un área de búsqueda hiperbólica con ciertos POI (*Points of Interest*) como la ilustrada en la siguiente figura, que representa los datos de entrada de un FA:

Figura LXXX: POI (Points of Interest) encerrados en un área de búsqueda

Como se pormenoriza en el capítulo referido a FA, el algoritmo explota en un punto, del que surgen *sparks* o centellas. Algunos de estos *sparks*, circulares en la figura, podrían atrapar el área que encierra un POI:

Figura LXXXI: Representación del FA y sparks susceptibles de hallar POI (Points of Interest)

Hasta el momento, el esquema mencionado se ajusta con precisión a la naturaleza básica de FA, donde, en caso de que un *spark* encuentre un POI, se lanzará un nuevo *firework* en ese punto para la próxima iteración. Ahora bien, en caso de que el área del *spark* encierre un POI, ¿cómo podría averiguarse el camino más rápido hacia el *firework*?

Es, precisamente al hilo de la pregunta anteriormente formulada, donde entra en juego ACO, puesto que un esquema híbrido entre dicha técnica y FA, permitiría utilizar un enfoque basado en el proceso de estigmergia para que las feromonas pertinentes indiquen el camino más directo al *firework*:

Figura LXXXII: Esbozo de la propuesta híbrida entre FA/ACO

Es reseñable que, una vez diseñada, la presente técnica híbrida será la razón de ser de una ampliación práctica en forma de sistema, que puede consultarse en el Anexo IV.

6.3 Mejora progresiva de sub-secciones de FA

Tal y como se ha especificado a lo largo de este documento, en la presente Tesis Doctoral se propone una mejora de FA, denominada IFA. La diferencia entre los diagramas de flujo de ambas entidades puede estudiarse seguidamente:

Figura LXXXIII: Diagrama de Flujo de FA (izquierda) y su contraparte en IFA (derecha)

Como puede apreciarse en la figura, existen nuevas fases en el algoritmo, así como ciertas modificaciones en las ya existentes. No obstante, la investigación podría ampliarse centrando el foco en las fases no modificadas hasta el momento; a saber:

- *Set off N fireworks at N locations*: El mecanismo de creación de fireworks, el cual sería lícito contemplar como una suerte de generación de una población inicial en cierto modo, podría rediseñarse a fin de acelerar la convergencia de IFA. Es especialmente reseñable [Khaji, 2014], estudio en el que se presenta un método heurístico para la generación de mejores poblaciones iniciales para métodos de corte evolutivo. En dicho estudio se demuestra cómo su método heurístico, aplicado a varias funciones de benchmarking (tal y como ocurre en la presente Tesis Doctoral), combina la naturaleza clásica y natural de ciertos eventos para generar una población inicial eficiente para tales heurísticas.
- *Gaussian explode for sparks*: Tal y como ocurre en la fase anterior, la metodología de explosión de las sparks puede ser rediseñada para

estudiar las posibles variaciones que este cambio pudiera producir en el rendimiento, en términos de *performance*, de IFA. La utilización de distintas distribuciones estadísticas (Exponencial, de Pareto, o Poisson, por citar algunas) puede constituir una interesante piedra angular para esta variación.

- *Optimal location found*: Dado que en un gran espacio de búsqueda esta fase podría ejecutarse millones de veces de forma ideal, la mejora del rendimiento en la misma podría impactar de forma tremendamente positiva en el rendimiento global de IFA. Por esta razón, esta fase se erige como un campo de mejora obligatorio en una línea futura de investigación que persiga la continuación de la actual mejora de IFA.

7-A. CONCLUSIONES

El espectro de la optimización algorítmica adscrita a la Computación Natural ha visto su elenco de entidades mastodónticamente aumentado, especialmente en la última década. Desafortunadamente, dicha ampliación no había sido paralela en su propia mejora. Por consiguiente, y tras presentar al lector un exhaustivo análisis del estado del arte de la Computación Natural y sus algoritmos principales (*Fireworks Algorithm*, *Genetic Algorithms*, *Ant Colony Optimization*, *Particle Swarm Optimization* o *Grammatical Evolution*, por citar algunos), en la presente Tesis Doctoral se ha llevado a cabo una mejora en el paradigma *Fireworks Algorithm* (FA), dando lugar al nacimiento de su versión mejorada, *Improved Fireworks Algorithm* (IFA). Es especialmente destacable que la base teórica sobre Natural Computing fruto de la presente Tesis Doctoral ha desembocado en la publicación de un artículo JCR circunscrito al cuartil Q2 en [Morales Lucas et al, 2018]. Por otro lado, se ha concebido y presentado el novedoso algoritmo ACORD (*Ant Colony systems Optimization applied to BNF grammars Rule Derivation*), hito que ha valido su publicación como otro artículo JCR en el cuartil Q2 en [de Mingo et al, 2019], consiguiendo dos publicaciones JCR Q2 fruto de la presente Tesis Doctoral.

Más en concreto, y respecto a IFA por un lado, en la presente Tesis Doctoral se ha demostrado que IFA presenta un rendimiento de más de un 40% mayor que FA en términos medios. No obstante, se recuerda que el *No Free Lunch Theorem* no es una excepción para IFA: Incluso los mejores modelos presentan en ocasiones un bajo rendimiento, mientras que modelos con una ínfima media de rendimiento en ocasiones se comportan de manera extremadamente eficiente. A pesar de esta constatación, cuando FA rinde mejor que IFA lo hace, en su mayoría, por un porcentaje mínimo, mientras que cuando IFA presenta más rendimiento que FA, lo hace por un porcentaje colosal. Funciones matemáticas de *benchmarking* como *Schaffer* demuestran que FA e IFA se desempeñan con similitud, mientras que *Axis-Parallel*, *Griewanks*, *Goldstein*, *Rastrigin* o *Ackley*, por citar algunas, demuestran que IFA disfruta de un rendimiento mucho mayor que FA, erigiéndose como una verdadera mejora del algoritmo. Una vez hallado tal enriquecimiento en términos algorítmicos, se ha descrito pormenorizadamente la metodología aplicada en pos de la consecución de dicho objetivo, analizando y discutiendo los resultados.

Por otro lado, en los aspectos referidos a las conclusiones extraídas del algoritmo ACORD, se refuerza la idea de que los algoritmos basados en ACO han demostrado ser una útil alternativa para solucionar problemas de *routing*. El algoritmo ACORD se aprovecha, precisamente, de esta característica para encontrar la solución a un problema de optimización, definido por una función de fitness en la que los mejores individuos pertenecen al lenguaje generado por una

gramática BNF. En ACORD, como se especifica en su epígrafe dedicado, puede apreciarse como los formícidos virtuales *caminan* a través de las reglas de derivación como si de un grafo se tratara, reforzando, por enésima vez, el propenso carácter de la Computación Natural hacia la combinación sinérgica. Esta idea, además, está inspirada por numerosos estudios sobre *Grammatical Evolution* y *Grammatical Swarm*, en su meta de separar genotipo de fenotipo. Por consiguiente, el espacio de búsqueda puede restringirse, y el conocimiento sobre el dominio del problema puede ser incorporado. Adviértase como es también posible examinar la convergencia en ciertas versiones de ACO, entendiendo como tal, por citar un ejemplo, su capacidad para encontrar el óptimo global en un tiempo finito. Es destacable, en la misma línea, que la primera evidencia de convergencia en ACO se realiza en [Gutjahr, 2000]. Una segunda evidencia surge, posteriormente, en los ACO basados en Teoría de Grafos, y seguidamente lo propio acontece en numerosos algoritmos relacionados con ACO, donde, como en la práctica totalidad de las metaheurísticas, es extremadamente difícil estimar la velocidad de convergencia a nivel teórico.

Englobando los razonamientos anteriores, se evidencia y concluye que ACORD abre una nueva vía a un campo de investigación en la Computación Natural en general y *Swarm Intelligence* en particular. Dicha vía, hasta ahora prácticamente inexistente, utiliza ACO para resolver problemas descritos en una gramática. Se concluye, además, que ACORD es capaz, a través de su capacidad de cambiar su comportamiento en función del problema, de centrar la búsqueda en áreas más útiles haciendo decrecer el espacio de búsqueda real. Es destacable que este deseable escenario acontece sin ni siquiera necesitar información de dominio sobre el problema en cuestión, y sin cambiar, de forma semántica, el espacio de búsqueda original. Los resultados experimentales presentados a lo largo de la presente Tesis Doctoral con respecto a ACORD demuestran, por tanto que ACO presenta un rendimiento superior al algoritmo *Grammatical Evolution* estándar en el problema *Santa Fe Trail*, independientemente de que éste último utilice un espacio de búsqueda semánticamente equivalente al del problema GP original, o de si usa el espacio de búsqueda sesgado del sistema de *Standard Grammatical Evolution* (BNF-O'Neil) utilizado en la literatura cuando el paradigma es presentado por O'Neil y Ryan en 2003. Por tanto, se infiere a modo de conclusión final de ACORD, que el algoritmo propuesto logra mejores resultados que el algoritmo de *Standard Grammatical Evolution* en términos no sólo de eficiencia, sino de efectividad a su vez.

A modo de colofón, numerosas líneas de ampliación, tanto teóricas como prácticas, se han incluido en la presente Tesis Doctoral, existiendo incluso un anexo orientado especialmente hacia la dotación de núcleos urbanos de inteligencia, a fin de que el término *smart city* sea una realidad palpable. Precisamente en la misma línea, se destaca que el espectro de aplicaciones que pueden extraerse de la Computación Natural para smart cities es tan inmenso

como útil, pues aún se necesita acometer un gran cambio para que los núcleos urbanos puedan considerarse ‘*smart*’. Un ejemplo de aplicación se expone, precisamente, en una de las dos publicaciones JCR Q2 fruto de esta Tesis Doctoral; [Morales Lucas et al, 2018]. De la misma forma que la Computación Natural mimetiza fenómenos naturales, es factible realizar una extrapolación a los diferentes sistemas que pertenecen a una ciudad, que puede verse como un enjambre de individuos o partículas, operando en los dominios de un sistema, en base a un enfoque *bottom-up*. Como puede intuirse, tal símil y la dualidad partícula-persona constituye una puerta abierta al amplio crecimiento del concepto *smart city* en base a la aplicación del Paradigma de Computación Natural, concepto objeto de [Chen et al., 2010].

Durante el anteriormente citado proceso, destaca la constante sensación de que el Paradigma de la Computación Natural se erige, por méritos propios, como un auténtico nicho de la optimización algorítmica, donde no sólo numerosas entidades otorgan un rendimiento excepcional, difícil de obtener hace una década, sino que además no han alcanzado su propia cota máxima de optimización y refinamiento, como se demuestra con *Fireworks Algorithm*. Por si tal hito fuera poco, ACORD demuestra, de forma paralela y como ya se ha mencionado, que el Paradigma de la Computación Natural brinda la excelente posibilidad de crear formidables sinergias entre las entidades que la componen, factor que refuerza la visión de que la Computación Natural es un auténtico nicho de optimización algorítmica.

En la misma línea, como se ha especificado en la presente Tesis Doctoral, y además de *Fireworks Algorithm*, la Computación Natural ha añadido técnicas tales como *Particle Swarm Optimization* (PSO) [Kennedy et al., 1995], que constituye un enfoque que pretende optimizar un problema a tenor de una estrategia meta-heurística, o, en otras palabras, en base al proceso iterativo de mejorar una solución genotípica hacia un fenotipo. Esta es, precisamente, la visión panorámica del *Genetic Algorithm*, también descrito al detalle en la presente Tesis Doctoral, estando presente durante todo momento el concepto de enjambre, que a su vez acontece nuevamente en *Ant Colony Optimization* (ACO). La amplia posibilidad de combinación sinercial a la que tiende la Computación Natural es, por ende, digna de subrayar, puesto que numerosos conceptos tienden a actuar siempre como aglutinantes a lo largo de este poderoso paradigma, que disfruta de las ventajas de la metaheurística; Alcanzar la solución en un periodo de tiempo ínfimo a costa de la carencia de obtención de intervalos de error.

Finalmente, y en un nivel de abstracción mayor, se destaca que en la presente Tesis Doctoral ha quedado expuesto como el *Massachusetts Institute of Technology* (MIT), cuando apuntó en Enero de 2018 a *Gene Therapy 2.0* y por ende a la Computación Natural como una de las tecnologías emergentes que

cambiarían el mundo en mayor medida, no estaba equivocado; no sólo esta tecnología es tan útil como versátil, sino que es capaz de abarcar campos tan dispares como la optimización matemática, el diseño de componentes microelectrónicos, la eficiencia energética, la medicina e incluso todos aquellos contenidos en el amplio espectro que los separa, ayudando a lograr la consecución del que debe ser el objetivo final, por ende, de cualquier proyecto de ingeniería: mejorar la calidad de vida de las personas.

7-B. CONCLUSIONS

Natural Computing spectrum has seen a massive growth on its entity count, especially over the last decade. Unfortunately, such growth was not parallel when it came to its own improvement. Thus, after presenting an exhaustive analysis about the State of the Art of Natural Computing and its main algorithms (i.e: *Fireworks Algorithm*, *Genetic Algorithms*, *Ant Colony Optimization*, *Particle Swarm Optimization* o *Grammatical Evolution*), the present Ph.D achieves, on the one hand, a tangible improvement for Fireworks Algorithm (FA), giving birth to its improved version, Improved Fireworks Algorithm (IFA). It is important to highlight that the theoretical basis exposed on this Ph.D. Thesis have deserved the publication of the JCR Q2 [Morales Lucas et al, 2018]. On the other hand, the novel ACORD (**Ant Colony systems Optimization applied to BNF grammars Rule Derivation**) Algorithm has been conceived and presented. Furthermore, this milestone has granted its JCR Q2 publication at [de Mingo et al, 2019], achieving two JCR Q2 publications out of this Ph.D. Thesis.

More concretely, this Ph.D demonstrates that IFA presents a more-than-40% better throughput than FA in average. Nevertheless, it is imperative to remind that the *No Free Lunch Theorem* is no exception for IFA: Even the best models present, occasionally, a low performance, while not-so-smooth mathematical models sometimes can behave in a extremely efficient way. In spite of this fact, when FA performs better than IFA, it does it by a minimal percentage, while when IFA performs better than FA, it does it in a colossal, overwhelming percentage. A few mathematical benchmarking functions, such Schaffer, demonstrate that FA and IFA share a similar performance on them, while Axis-Parallel, Griewanks, Goldstein, Rastrigin or Ackley, to remind a few of them, demonstrate that IFA experiences a much greater performance than FA. This blatancy means that IFA stands up as a true improvement in the algorithm.

Once such algorithmic enrichment has been achieved, this document describes, in a fine-grained way, the methodology applied to reach that goal, in order to analyze and discuss the obtained results.

Conspicuously, regarding those aspects referred to the conclusions extracted from ACORD Algorithm, the fact that ACO-based algorithms are a useful alternative in order to solve routing related problems gets reinforced. ACORD Algorithm takes advantage of this feature to find the solution to an optimization problem in which candidates belong to a language which is generated by a BNF grammar. Virtual ants walk through the derivation rules as if it were a graph. This idea is inspired by the works of Grammatical Evolution and Grammatical Swarm, based on the desire to separate genotype from phenotype. Therefore, the search space can be restricted, and knowledge of the domain of the problem can be

incorporated. It is also possible to test convergence in some versions of the ant colony algorithm (i. e. it is able to find the overall optimal in finite time). The first evidence of an ant colony convergence algorithm was made in [Gutjahr, 2000], the graph-based ant system algorithm, and then algorithms for Ant Colony Systems. However, like most metaheuristics, it is very difficult to estimate the theoretical speed of convergence.

To sum up, it is remarkable that this new algorithm opens the way to a new branch of research in Swarm Intelligence, which until now has been almost non-existent, using ant colony algorithms to solve problems described by grammar. Proposed algorithm is able through its conditional behaviour-switching feature to focus the search in more useful areas by decreasing the actual search space without using domain knowledge of the problem in question and without changing semantically the original search space. The experimental results presented in this Ph.D show that ant colonies outperforms the standard GE algorithm in the Santa Fe Trail problem whether the later uses a search space semantically equivalent with that of the original GP problem, or whether it uses the biased search space of the standard Grammatical Evolution system (BNF-O'Neill) used in the Grammatical Evolution literature as was first presented by O'Neill and Ryan in 2003. Namely, proposed algorithm achieves better results than the standard Grammatical evolution algorithm in terms of both efficiency and effectiveness.

As a brochure, plentiful ampliation lines, of both theoretical and practical idiosyncrasy, were shown this Ph.D, even defining a dedicated annex especially oriented to the smart cities spectrum, chasing urban cores intelligence growth, willing to make the aforementioned smart city concept a tangible reality. Precisely in the same line, it is mandatory to remind that the set of applications that are to be extracted from Natural Computing spectrum for smart cities is humongous, just imagination-bounded. This factor turns smart cities in an actual business niche, for a massive change in urban cores to be called *intelligent* is to be done. An example of an application is indeed described in one of the two JCR Q2 publications extracted from this Ph.D. Thesis; [Morales Lucas et al, 2018]. Along the same line as Natural Computing mimetizes natural phenomena, it is possible to extrapolate the different systems that compound a city as we know it, that may be seen as a swarm of individuals or particles, operating among a system bounds, following a bottom-up schema. As it might be surmised, this simile and the particle-person duality constitutes an open door to an exponential growth in the *smart city* concept thanks to the Natural Computing paradigm, as seen in [Chen et al., 2010].

During the aforementioned process, the constant feeling about the fact that Natural Computing paradigm stands up, by its own merits, as a genuine field of investigation for algorithmic optimization, where many entities are throwing exceptional results and performances. Even though these results, unthinkable a decade ago, are there, this Ph.D and the works referenced on it demonstrate that

Natural Computing algorithms have not reached its own peak when it comes to optimization and refinement, as FA-IFA challenge exhibited in the present document. Just in case this fact was not enough, ACORD proves, in a parallel way and as aforementioned, that Natural Computing paradigm brings the astonishing chance of creating formidable synergies among the entities that make it up, which reinforces the vision of Natural Computing as an indubitable niche when it comes to algorithm optimization field.

Along the same line, as it has been stated during the present Ph.D, and apart from Fireworks Algorithm, Natural Computing added techniques such Particle Swarm Optimization (PSO) [Kennedy et al., 1995]. PSO constitutes an approach willing to optimize a problem due to a metaheuristic strategy, or, in other words, following the iterative process of improving a genotypic solution towards a phenotype. This is actually the global overview (also thoroughly described along this document) associated to Genetic Algorithms, being present during the whole concept of *swarm*. Such concept appears once again at Ant Colony Optimization (ACO). Please note that, on the one hand, the enormous possibility of sinergial combination among Natural Computing algorithms is remarkable, as long as many concepts tend to act as binders through this powerful paradigm. On the other hand, Natural Computing enjoys metaheuristic benefits and privileges; reaching the solution in an insignificant amount of time at the expense of the lack of information about error bounds.

Finally, and at a superior level of abstraction, it is truly important to note that something else has been exposed in the present Ph.D: When Massachusetts Institute of Technology (MIT) pointed to Gene Therapy 2.0 and thus to Natural Computing in January 2018 as a world-changing technology, they weren't wrong. This paradigm and its technology are not only as useful as versatile, but capable as well to embrace fields as diverse as mathematical optimization, microelectronic components design, energetic efficiency, medicine, and even those comprehended in between the mastodonic spectrum separating them, helping to achieve what shall be the final objective of any engineering project; improve people's quality of life.

Referencias Bibliográficas

[Abramowitz et al., 1983] Abramowitz, M., Stegun, I. (1983) 26-1. "Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables." Applied Mathematics Series. Washington D.C.; New York: United States Department of Commerce, National Bureau of Standards; Dover Publications. p. 940. ISBN 978-0-486-61272-0. LCCN 64-60036. MR 0167642. LCCN 65-12253. 1983.

[Acampora et al., 2016] G. Acampora and A. Vitiello, "jMeme: a Java Library for Designing Competent Memetic Algorithms," 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Vancouver, Canada, 24-29 July 2016. 2016.

[Ackley, 1987] Ackley, D. H. (1987) "A connectionist machine for genetic hillclimbing", Kluwer Academic Publishers, Boston MA. 1987.

[Adorio et al., 2005] Adorio, P., Diliman, P. (2005) "MVF - Multivariate Test Functions Library in C for Unconstrained Global Optimization". Springer-Verlag. 2005.

[Akbari, 2010] Akbari, Z. (2010). "A multilevel evolutionary algorithm for optimizing numerical functions IJIEC 2" (2011): 419–430 [1]. 2010.

[Alfonseca et al., 2015] Alfonseca, M., Gil, F., (2015). "Evolving a predator-prey ecosystem of mathematical expressions with grammatical evolution". Springer Verlag. 2015.

[Amdahl, 1967] Amdahl, G. (1967). "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities". AFIPS Conference Proceedings (30): 483–485. doi:10.1145/1465482.1465560. 1967.

[Amsterdam, 2018] Amsterdam Smart City. (2018). "Amsterdam Smart City". [online] Available at: <http://amsterdamsmartcity.com/projects/theme/label/smart-mobility> [Accessed 16 Apr. 2017].

[Areibi et al., 2004] Areibi, S., Yang, Z. (2004). "Effective memetic algorithms for VLSI design automation = genetic algorithms + local search + multi-level clustering". Evolutionary Computation. MIT Press. 12 (3): 327–353. PMID 15355604. doi:10.1162/106365604. 2004.

[Babaei, 2013] Babaei, M. (2013) "A novel text and image encryption method based on chaos theory and DNA computing". Natural Computing 12 (1), 101-107. 2013.

[Bachmann, 1894] Bachmann, P. (1894). "Analytic Number Theory". Web Of Science. p. 34-98. 1894.

[Baker, 1985] Baker, J. (1985). "Adaptive Selection Methods for Genetic Algorithms". Proceedings of the 1st International Conference on Genetic Algorithms. 0-8058-0426-9, pp.101-111. 1985.

[Banks et al., 1981] Alec, B., Vincent, J., Anyakoha, C. "A review of particle swarm optimization". MIT Press, Cambridge, MA. 1981.

[Banzhaf et al., 1998] Banzhaf, W., Nordin, P., Keller, R., Francone, F. (1998). "Genetic Programming – An Introduction". San Francisco, CA: Morgan Kaufmann. ISBN 978-1558605107. 1998.

[Bard, 1992] Bard, J. "Morphogenesis: The Cellular and Molecular Processes of Developmental Anatomy". Developmental and Cell Biology Series. 1992.

[Bath, 2018] University of Bath, (2018). "Grammatical Evolution in Python". Available at: <http://www.cs.bath.ac.uk/~mdv/courses/CM30082/projects.bho/2005-6/painter-t-dissertation-2005-6.pdf> [Accessed 17 Feb. 2018].

[Bauer, 2001] Bauer, H. (2001). "Measure and Integration Theory". Berlin: de Gruyter. ISBN 3110167190. 2001.

[Bellman, 2016] Bellman, R. (2016). "Adaptive Control Processes". Princeton, NJ: Princeton University Press. 2016.

[Beni, 2005] Beni G (2005) From swarm intelligence to swarm robotics. In: Proceedings of the 2004 International Conference on Swarm Robotics, Springer-Verlag, Berlin, Heidelberg, SAB'04, pp 1–9, DOI 10.1007/978-3-540-30552-1 1. 2005.

[Beni et al., 1989] Beni, G., Wang, J. (1989) "Swarm Intelligence in Cellular Robotic Systems, Proceed". NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26–30. 1989.

[Bennett, 1996] Bennett, S. (1996). "A brief history of automatic control" (PDF). IEEE Control Systems Magazine. IEEE. 16 (3): 17–25. 1996.

[Berg, 2001] Berg. V. (2001) "An Analysis of Particle Swarm Optimizers". Faculty of Natural and Agricultural Science. University of Pretoria. Pretoria. 2001.

[Berkeley, 2018] Evolution.berkeley.edu. (2018). "Early Concepts of Evolution: Jean Baptiste Lamarck". [online] Available at: http://evolution.berkeley.edu/evolibrary/article/history_09 [Accessed 5 Mar. 2017].

[Bies et al., 2006] Bies, R., Muldoon, M., Pollock, B., Manuck, S., Smith, G., Sale, M. (2006). "A Genetic Algorithm-Based, Hybrid Machine Learning Approach to Model Selection". *Journal of Pharmacokinetics and Pharmacodynamics* (Netherlands: Springer). 2006.

[Blum et al., 2008] Blum, C., Li, X. "Swarm Intelligence in Optimization". Springer Verlag, 2008.

[Bonabeau et al., 1999] E. Bonabeau, M. Dorigo et G. Theraulaz, 1999. "Swarm Intelligence: From Natural to Artificial Systems", Oxford University Press. ISBN 0-19-513159-2. 1999.

[Bonyadi et al., 2014] Bonyadi, M., Michalewicz, Z. and Li, X. (2014). "An analysis of the velocity updating rule of the particle swarm optimization algorithm". *Journal of Heuristics*, 20(4), pp.417-452. 2014.

[Bonyadi et al., 2014] Bonyadi, M. and Michalewicz, Z. (2014). "A locally convergent rotationally invariant particle swarm optimization algorithm". *Swarm Intelligence*, 8(3), pp.159-198. 2014.

[Bozorgirad et al., 2016] Bozorgirad, M.A., Logendran, R. (2016) *International Journal of Advanced Manufacturing Technology* 83: 1135. 2016.

[Brassard et al., 1996] Brassard, G. and Bratley, P. (1996) "Fundamental of Algorithmics", Prentice-Hall. p. 22-59. 1996.

[Bratton et al., 2007] D. Bratton and J. Kennedy (2007). "Defining a Standard for Particle Swarm Optimization". *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*, pp. 120-127. 2007.

[Bäck, 1995] Bäck, T. (1995). "Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms". Oxford: Oxford University Press. p. 328-352. 1995.

[Campelo, 2018] EC-Bestiary. (2018). "Evolutionary Computation Bestiary". [online] Available at: <http://fcampelo.github.io/EC-Bestiary/> [Accessed 5 Mar. 2018].

[Caruana et al., 2006] Caruana R., Niculescu-Mizil, A. (2006) "An Empirical Comparison of Supervised Learning Algorithms", Department of Computer Science, Cornell University, USA. p 3-8. 2006.

[Casella et al., 2001] Casella,G; Berger, R. (2001). "Statistical Inference" (2nd ed.). Duxbury. 2001.

[CEOE, 2015] Informe de la Confederación Española de Organizaciones Empresariales (CEOE) "Acciones Prioritarias para el Desarrollo de las Smart Cities en España". Comité de Smart Cities Madrid. 2015.

[Cha et al., 2009] Cha, S., Tappert, C. (2009). "A Genetic Algorithm for Constructing Compact Binary Decision Trees". Journal of Pattern Recognition Research. 2009.

[Chen et al, 2011] Chen, X. S.; Ong, Y. S.; Lim, M. H.; Tan, K. C. (2011). "A Multi-Facet Survey on Memetic Computation". IEEE Transactions on Evolutionary Computation. 15 (5): 591–607. doi:10.1109/tevc.2011.2132725. 2011.

[Chen et al., 2010] Chen, H. Zhu, Y. Hu, K. (2010). "Discrete and continuous optimization based on multi-swarm coevolution". Natural Computing, 9(5).659-682. 2014.

[Chen et al., 2014] Chen, HL. Doty, D. Soloveichik, D. (2014) "Deterministic function computation with chemical reaction networks". Natural Computing 13 (4), 517-534. 2014.

[Cormen et al., 2001] Cormen, H., Leiserson, E., Rivest, L., Stein, C. (2001) "Introduction to Algorithms". Second Edition. MIT Press and McGraw-Hill.2-2, p.40. 2001.

[Cran, 2018] Anon, (2018). "Introduction to Genetic Evolution" [PDF] Available at: <https://cran.r-project.org/web/packages/gramEvol/vignettes/ge-intro.pdf> [Accessed 2 Jul. 2017].

[Crols, 2017] Crols, T., White, R., Uljeel, I., Poelmans, L., Engelen, G., Canters, F.. "High-resolution simulations of population-density change with an activity-based cellular automata land-use model". Proceedings of Geo-Computation, 2017.

[Dam, 2013] Dam, W. (2013) "Implausible consequences of superstrong nonlocality". Natural Computing 12 (1), 9-12. 2013.

[Darwin et al., 1859] Livezey, R. and Darwin, C. (1859). "On the Origin of Species by Means of Natural Selection". American Midland Naturalist, 49(3), p.937. 1953.

[Dawkins, 2000] Dawkins, E. (2000) "The Selfish gene". Barcelona: Editorial Salvat.

[Merz et al, 2002] Merz, P.; Zell, A. (2002). Clustering Gene Expression Profiles with Memetic Algorithms. Parallel Problem Solving from Nature — PPSN VII. Springer. pp. 811–820. doi:10.1007/3-540-45712-7_78. 2000.

[de Campos et al., 2002] de Campos, L., Fernández-Luna, J., Gámez, J. and Puerta, J. "Ant colony optimization for learning Bayesian networks. International Journal of Approximate Reasoning", 31(3), pp.291-311. 2002.

[de Castro et al., 2002] De Castro, J. Timmis, N. "Artificial Immune Systems: A New Computational Intelligence Approach". Springer Verlag, 2002.

[de Castro, 2007] de Castro, L. N., "Fundamentals of Natural Computing: Basic Concepts", Algorithms, and Applications, CRC Press, 2006.

[de Mingo et al., 2009] de Mingo, L., Gómez, N. Garitagoitia, J. Martinez, V. Calvo, J. (2009) "Grammatical Swarm and Particle Swarm Optimization models applied to Neural Network learning and topology definition". 2009.

[de Mingo et al, 2019] de Mingo López, L.F.; Gómez Blas, N.; Morales Lucas, C.: Ant Colony systems Optimization applied to BNF grammars Rule Derivation (ACORD Algorithm). Soft Computing. ISSN: 1432-7643 (Print) 1433-7479 (Online). 2019. (Pendiente de publicación).

[Dempsey et al., 2009] Dempsey, I., O'Neill, M. and Brabazon, A. "Foundations in grammatical evolution for dynamic environments". Berlin: Springer. 2009.

[Dixon et al., 1978] Dixon, C., Szego, W. (1978). "The global optimization problem: an introduction. Towards global optimization", 2, p. 1-15. 1978.

[Dixon et al., 1998] Dixon, L., Szego, G. (1978). "The global optimization problem: an introduction." Towards global optimization, 2, 1-15. 1978.

[Dorigo, 1992] Dorigo, M. (1992). "Optimization, Learning and Natural Algorithms", PhD thesis, Politecnico di Milano, Italy. 1992.

[Dorigo et al., 1996] M. Dorigo, V. Maniezzo & A. Colorni, 1996. Ant System: "Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man, and Cybernetics"—Part B, 26 (1): 29–41. 1996.

[Dorigo et al., 1997] Dorigo, M., Gambardella, L. (1997). "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem". IEEE Transactions on Evolutionary Computation, 1 (1): 53–66. 1997.

[Dorigo et al., 1999] Dorigo, M., Di Caro, G., Gambardella, L. (1999). "Ant Algorithms for Discrete Optimization". Artificial Life, 5 (2): 137–172. 1999.

[Dorigo et al., 2004] Dorigo, M., Stützle, T. (2004). "Ant Colony Optimization", MIT Press. ISBN 0-262-04219-3. 2004.

[Eiben, 1994] Eiben, A. (1994). "Genetic algorithms with multi-parent recombination." PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: 78–87. ISBN 3-540-58484-6. 1994.

[Elbeltagi et al., 2005] Elbeltagi, A. Hegazy, T. Grierson, D. (2005). "Comparison among five evolutionary-based optimization algorithms". Science Direct (3) doi:<https://doi.org/10.1016/j.aei.2005.01.004>. 2005.

[Engelbrecht, 2013] Engelbrecht AP (2013) Particle swarm optimization: Global best or local best? In: 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence, pp 124–135, DOI 10.1109/BRICS-CCI-CBIC.2013.31. 2013.

[Eurocities, 2018] Eurocities.eu. (2018). mobility. [online] Available at: <http://www.eurocities.eu/eurocities/forums/mobility&tpl=home> [Accessed 11 May 2017].

[Fernández-Márquez et al., 2013] Fernández-Márquez, JL. Serugendo, GDM. Montagna, S, Viroli, M. Arcos, JL. (2013) "Description and composition of bio-inspired design patterns: a complete overview." Natural Computing 12 (1), 43-67. 2013.

[Flir, 2018] FLIR Systems, I. (2018). FLIR Systems | "Thermal Imaging, Night Vision and Infrared Camera Systems." [online] Flir.es. Available at: <http://www.flir.es/traffic> [Accessed 2 Dec. 2015].

[Fogel, 1998] Fogel, D. "Evolving computer programs." In Evolutionary Computation: The Fossil Record, D. Fogel, Ed. MIT Press, 1998, ch. 5, pp. 143–144. 1998.

[Fogel, 2003] Fogel, D. "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence" (3rd ed.). Piscataway, NJ: IEEE Press. ISBN 978-0471669517. 2003.

[Fraser, 1957] Fraser, A. (1957). "Simulation of Genetic Systems by Automatic Digital Computers." I. Introduction. Australian Journal of Biological Sciences. 1957.

[Gamma, 2011] Gamma, E. "Design patterns". Boston [u.a.]: Addison-Wesley. 2011.

[Gan, 2010] Gan R, Guo Q, Chang H, Yi Y (2010) Improved ant colony optimization algorithm for the traveling salesman problems. Journal of Systems Engineering and Electronics 21(2):329–333, DOI 10.3969/j.issn.1004-4132.2010.02.025. 2010.

[Garg, 2015] Garg, H. (2015) "A hybrid PSO-GA algorithm for constrained optimization problems." Applied Mathematics and Computation. Elsevier. (1), 2-6. 2015.

[Georgiou, 2011] Georgiou L, Teahan WJ (2011) Constituent Grammatical Evolution. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two, AAAI press, IJCAI'11, pp 1261–1268, DOI 10.5591/978-1-57735-516-8/ijcai11-214. 2011.

[GEVA, 2018] Ncra.ucd.ie. (2018). UCD NCRA - Software - GEVA. [online] Available at: <http://ncra.ucd.ie/GEVA.html> [Accessed 14 Jun. 2017].

[Gheorghe et al., 2012] Zhang, G. Gheorghe, M. Li, Y. (2012) "A membrane algorithm with quantum-inspired subalgorithms and its application to image processing." Natural Computing 11 (4), 701-717. 2012.

[Gob, 2018] Ley Orgánica 15/1999, Estado, J. 2011, V. (2018). Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal. [online] Noticias Jurídicas. Available at: http://noticias.juridicas.com/base_datos/Admin/lo15-1999.html [Accessed 20 Jul. 2017].

[Goldberg, 1989] Goldberg, D. (1989). "Genetic Algorithms in Search, Optimization and Machine Learning." Reading, MA: Addison-Wesley Professional. ISBN 978-0201157673. 1989.

[Goldberg, 1991] Goldberg, D. (1991). "The theory of virtual alphabets. Parallel Problem Solving from Nature", Lecture Notes in Computer Science 496: 13–22. doi:10.1007/BFb0029726. 1991.

[Goldberg, 2002] Goldberg, D. (2002). "The Design of Innovation: Lessons from and for Competent Genetic Algorithms." Norwell, MA: Kluwer Academic Publishers. ISBN 978-1402070983. 2002.

[Grammatical, 2018] Grammatical-evolution.org. (2018). www.grammatical-evolution.org. [online] Available at: <http://www.grammatical-evolution.org/> [Accessed 22 Nov. 2016].

[Grendeau, 1998] Gendreau M, Laporte G, Semet F (1998) A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research* 106(2-3):539–545, DOI 10.1016/S0377-2217(97)00289-0. 1998.

[Gutjahr, 2000] Gutjahr WJ (2000) A graph-based ant system and its convergence. *Future Generation Computer Systems* 16(8):873–888, DOI 10.1016/S0167-739X(00)00044-3. 2000.

[Haight, 1967] Haight, F. (1967). "Handbook of the Poisson Distribution." New York: John Wiley & Sons. 1967.

[Hart, 1994] Hart W. E. (1994). "Adaptive Global Optimization with Local Search". University of California. 1994.

[Heppner et al., 1990] Heppner, F., U. Grenander (1990). "A stochastic nonlinear model for coordinated bird flocks." S. Krasner, Ed., *The Ubiquity of Chaos*. AAAS Publications, Washington, DC. 1990.

[Hoang et al., 2012] Hoang, T., Bhanu, B. "Zombie Survival Optimization: A swarm intelligence algorithm inspired by zombie foraging." *Pattern Recognition (ICPR), 2012 21st International Conference on IEEE*, 2012.

[Holland, 1975] Holland, J. (1992). "Adaptation in natural and artificial systems." Cambridge, Mass.: MIT Press. 1992.

[Holland, 1992] Holland, J. (1992). "Adaptation in Natural and Artificial Systems." MIT Press, Cambridge, MA. 1992.

[Horizonte, 2018] Eshorizonte2020.es. (2018). ¿Qué es Horizonte 2020? / Horizonte2020. [online] Available at: <http://www.eshorizonte2020.es/que-es-horizonte-2020> [Accessed 30 Abr. 2017].

[Hornby et al., 1999] Hornby, G., Lohn, J. and Linden, D. (1999). "Computer-Automated Evolution of an X-Band Antenna for NASA's Space Technology 5 Mission." *Evolutionary Computation*, 19(1), pp.1-23. 1999.

[Hugosson, 2007] Hugosson J, Hemberg E, Brabazon A, O'Neill M (2007) An investigation of the mutation operator using different representations in grammatical evolution. In: 2nd International Symposium "Advances in Artificial Intelligence and Applications", Wisla, Poland, vol 2, pp 409–419, URL <http://www.proceedings2007.imcsit.org/pliks/45.pdf>. 2007

[Ichimura et al., 1998] Ichimura, T.; Kuriyama, Y. (1998). "Learning of neural networks with parallel hybrid GA using a royal road function." IEEE International Joint Conference on Neural Networks. 2. New York, NY. pp. 1131–1136. 1998.

[Janecek et al., 2011] Janecek, A. and Tan, Y. (2011). "Swarm Intelligence for Non-Negative Matrix Factorization." International Journal of Swarm Intelligence Research, 2(4), pp.12-34. 2011.

[Janikow et al., 1991] Janikow, C., Michalewicz, Z. (1991). "An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms." Proceedings of the Fourth International Conference on Genetic Algorithms: 31–36. 1991.

[Johannes, 2003] Johannes, V. (2003). "The Statistical Mechanics of Financial Markets." Springer-Verlag. p. 124. ISBN 3-540-00978-7. 2003.

[Jovanovic, 2010] Jovanovic R, Tuba M, Simian D (2010) Comparison of different topologies for island-based multi-colony ant algorithms for the minimum weight vertex cover problem. W.Trans on Comp 9(1):83–92. 2010.

[Karkavitsas et al., 2011] Karkavitsas G., Tsihrintzis, G. (2011). "Automatic Music Genre Classification Using Hybrid Genetic Algorithms". Intelligent Interactive Multimedia Systems and Services. Springer. 11: 323–335. doi:10.1007/978-3-642-22158-3_32. 2011.

[Kennedy, 1997] Kennedy, J. (1997). "The particle swarm: social adaptation of knowledge". Proceedings of IEEE International Conference on Evolutionary Computation. 1997.

[Kennedy, 2006] J. Kennedy (2006). "Handbook of Nature-Inspired and Innovative", Springer, Chapter 6, pp. 187-219. 2006.

[Kennedy, 2006] Kennedy J, Mendes R (2006) Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 36(4):515–519, DOI 10.1109/TSMCC.2006.875410. 2006.

[Kennedy et al., 1995] J. Kennedy and R. Eberhart(1995), "Particle Swarm Optimization." Proceedings of the 1995 IEEE International Conference on Neural Networks (Perth, Australia): IEEE Service Center, Piscataway, NJ, IV, pp. 1942-1948. 1995.

[Kennedy et al., 2001] Kennedy, J., Eberhart, R.C. (2001). "Swarm Intelligence". Morgan Kaufmann, ed. 2001.

[Khaji, 2014] Khaji, E. (2014) "A Heuristic Method to Generate Better Initial Population for Evolutionary Methods". 1-7. 2014.

[Klug et al., 2000] Klug, W. S. & Cummings, "M. R. Concepts of Genetics". Prentice Hall, 6.a edición, 2000.

[Knuth, 1998] Knuth, D. (1998). "Sorting and Searching. The Art of Computer Programming." 3 (2nd ed.) Reading, MA: Addison-Wesley Professional. 1998.

[Kollin, 2017] Kollin F, Bavey A (2017) Ant colony optimization algorithms: Pheromone techniques for tsp. Tech. rep., KTH, School of Computer Science and Communication (CSC). 2017.

[Koza,1989] Koza, J. R. Hierarchical genetic algorithms operating on populations of computer programs. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89 (Detroit, MI, USA, 20-25 Aug. 1989), N. S. Sridharan, Ed., vol. 1, Morgan Kaufmann, pp. 768–774. 63. Koza, J. R. A genetic approach to the truck. 1989.

[Koza, 1992] Koza, J. "Genetic Programming: On the Programming of Computers by Means of Natural Selection". Statistics and Computing, 4(2). 1992.

[Kromer, 2014] Krömer P, Platoš J, Snášel V (2014) Nature-inspired metaheuristics on modern gpus: State of the art and brief survey of selected algorithms. Int J Parallel Program 42(5):681–709, DOI 10.1007/s10766-013-0292-3. 2014.

[Kushchu, 2002] Kushchu I (2002) Genetic programming and evolutionary generalization. Trans Evol Comp 6(5):431–442, DOI 10.1109/TEVC.2002.805038. 2002

[Laguna et al., 2002] Laguna, M., Marti, R. (2002) "Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions." Prentice-Hall. p. 35-69. 2002.

[Laguna et al., 2002] Laguna, M., Marti, R. "Experimental Testing of Advanced Scatter". Search Designs for Global Optimization of Multimodal Functions. 2002.

[Langdon, 1998] Langdon WB, Poli R (1998) Better trained ants for genetic programming. Tech. Rep. CSRP-98-12, University of Birmingham. 1998.

[Lamarck, 1801] Lamarck, J. (1801). "Système des animaux sans vertèbres. A Paris: Chez l'auteur ...". Deterville. 1801.

[Larrañaga, 1996] Larrañaga, P., "Learning Bayesian Network Structures by searching for the best ordering with genetic algorithms.", IEEE Transactions on systems, man and cybernetics, Vol 26, No. 4, 1996.

[Lee et al., 1999] Lee, L., Zhang, Y. (2010). "Linear Projection-based Non-negative Matrix Factorization". Acta Automatica Sinica, 36(1), pp.23-39. 2010.

[Lee, 2008] Lee ZJ, Su SF, Chuang CC, Liu KH (2008) Genetic algorithm with ant colony optimization (ga-aco) for multiple sequence alignment. Appl Soft Comput 8(1):55–78, DOI 10.1016/j.asoc.2006.10.012

[Levitin, 2002] Anany V. Levitin. (2002) "Introduction to the Design and Analysis of Algorithms". Addison Wesley. 2002.

[Liacs, 2016] Ant Colony Optimization. Anon. [offline] Available at: <http://natcomp.liacs.nl/SWI/papers/ant.colony.optimization/The.ant.colony.optimization.meta.heuristics.pdf> [Accessed 28 Abr.2016]

[Liu et al., 2004] R. Liu, H. Du, L. Jiao. "Immunity poly-clonal strategy." Journal of Evolutionary Computing. Res. Dev. 41(4), 571–576. 2004.

[Liu, 2009] Liu F, Zeng G (2009) Study of genetic algorithm with reinforcement learning to solve the tsp. Expert Syst Appl. 36(3):6995–7001, DOI 10.1016/j.eswa.2008.08.026. 2009.

[Lones, 2014] Lones MA (2014) Metaheuristics in nature-inspired algorithms. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, ACM, New York, NY, USA, GECCO Comp '14, pp 1419–1422, DOI 10.1145/2598394.2609841. 2014.

[Lyon, 2018] Lyon.smartcityapphack.com. (2018). "Challenges – Lyon Smart City App Hack." [online] Available at: <http://lyon.smartcityapphack.com/challenge/> [Accessed 14 Jan. 2018].

[Madrid, 2018] Comunidad de Madrid, (2018). "Dossier sobre el Metro de Madrid". [ebook] Available at: http://www.metromadrid.es/comun/noticias/pdf/Dossier_90_axos_de_Metro.OK.pdf [Accessed 26 Sep. 2016].

[Malaga, 2018] Ayuntamiento de Málaga, (2018). [online] Available at: <http://www.malagavalley.com/index.php/es/malaga-smart/transporte-publico> [Accessed 31 Jan. 2016].

[Manalu, 2018] Manalu, G. (2018). "Particle Swarm Optimization (PSO) Sample Code using Java". [online] Gandhi Manalu. Available at: <https://gandhim.wordpress.com/2010/04/04/particle-swarm-optimization-pso-sample-code-using-java/> [Accessed 1 May. 2016].

[Manville et al., 2014] Manville, C., Cochrane, G. and Cave, J. (2014). "Mapping smart cities in the EU." Luxembourg: EUR-OP. 2014.

[Meer, 2007] Meer, K. "Simulated annealing versus metropolis for a tsp instance." Information Processing Letters 104(6):216–219, DOI 10.1016/j.ipl.2007.06.016. 2007.

[Mellema, 2008] Mellema, G. "The Oxford Companion to Philosophy". History and Philosophy of Logic, 2. 2008.

[Metro, 2018] Play.google.com. (2018). "Metro de Madrid" (App.). [online] Available at: <https://play.google.com/store/apps/details?id=es.metromadrid.metroandroid&hl=es> [Accessed 15 Feb. 2018].

[Miller et al., 1985] Miller, B. and Goldberg, D. (1985). "Genetic algorithms, tournament selection and the effects of noise." Urbana, Ill. 1985.

[Miranda, 2016] Miranda, P. Prudêncio, R. "Tree-Based Grammar Genetic Programming to Evolve Particle Swarm Algorithms," 5th Brazilian Conference on Intelligent Systems (BRACIS), Recife, 2016, pp. 25-30. doi: 10.1109/BRACIS.2016.016. 2016.

[Mirjalili et al., 2016] Mirjalili, S. Lewis, A. (2016) "The Whale Optimization Algorithm." Elsevier. School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane, Australia. 2016.

[Molga et al., 2005] Molga, M., & Smutnicki, C. (2005). [ebook] Available at: <http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf> [Accessed 15 Sep. 2017].

[Molga et al., 2005] Molga, M., Smutnicki, C. (2005) “Test functions for optimization needs.” Oxford University Press. 2005.

[Moore, 2008] Moore, J., Andrews, P., Barney, N., and White, B. “Development and evaluation of an open-ended computational evolution system for the genetic analysis of susceptibility to common human diseases”. (2008) Lecture Notes in Computer Science 4973, 129–140. 2008.

[Morales Lucas et al, 2018] Morales Lucas, C.; de Mingo López, L.F.; Gómez Blas, N. “Natural Computing Applied to the Underground System: A Synergistic Approach for Smart Cities”. (2018) *Sensors* 2018, 18, 4094. 2008.

[Moscato, 1989] Moscato, P. (1989). “On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms.” Caltech Concurrent Computation Program. 1989.

[Moslehi, 2014] Moslehi G, Khorasanian D (2014) A hybrid variable neighborhood search algorithm for solving the limited-buffer permutation flow shop scheduling problem with the makespan criterion. *Comput Oper Res* 52(PB):260–268, DOI 10.1016/j.cor.2013.09.014

[Mullin, 2018] Mullin, E. (2018). “How did gene therapy go from experimental disasters to wondrous cures?”. [online] MIT Technology Review. Available at: <https://www.technologyreview.com/s/603498/10-breakthrough-technologies-2017-gene-therapy-20/> [Accessed 2 Oct. 2017].

[NCRA, 2018] Ncra.ucd.ie. (2018). “Grammatical Swarm.” [online] Available at: http://ncra.ucd.ie/papers/ncj2006_gswarm.pdf [Accessed 28 Nov. 2017].

[Neill et al., 2018] Neill, M., Brabazon, A. and Adley, C. (2018). “The automatic generation of programs for classification problems with grammatical swarm.” [online] Academia.edu. Available at: http://www.academia.edu/2649822/The_automatic_generation_of_programs_for_classification_problems_with_grammatical_swarm:The_automatic_generation_of_programs_for_classification_problems_with_grammatical_swarm [Accessed 13 Feb. 2018].

[Neumann, 2008] Neumann F, Sudholt D, Witt C (2008) Rigorous analyses for the combination of ant colony optimization and local search. In: Ant Colony Optimization and Swarm Intelligence, 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008. Proceedings, pp 132–143, DOI 10.1007/978-3-540-87527-7 12. 2008.

[O'Neill, 2010] O'Neill, M., Vanneschi, L., Gustafson, S. and Banzhaf, W. (2010). Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4), pp.339-363. 2010.

[O'Neill, 2001] O'Neill M, Ryan C, Nicolau M (2001) Grammar defined introns: An investigation into grammars, introns, and bias in grammatical evolution. In: Spector L, Goodman ED, Wu A, Langdon WB, Voigt HM, Gen M, Sen S, Dorigo M, Pezeshk S, Garzon MH, Burke E (eds) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, Morgan Kaufmann, San Francisco, California, USA, pp 97–103. 2001.

[O'Neill, 1999] O'Neill M, Ryan C (1999) Under the hood of grammatical evolution. In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, GECCO'99, pp 1143–1148. 1999.

[Olariu et al., 2005] Olariu S., Zomaya A. Y., "Handbook of Bioinspired Algorithms and Applications.", Chapman & Hall/CRC, 2005.

[Oldenhuis, 2009] Oldenhuis, R. (2009) "Many test functions for global optimizers." Mathworks. 2009.

[Ong, 2004] Ong Y. S. & Keane A. J. (2004). "Meta-Lamarckian learning in memetic algorithms". *IEEE Transactions on Evolutionary Computation*. 8 (2): 99–110. doi:10.1109/TEVC.2003.819944. 2004.

[Oplatková, 2006] Oplatková Z, Zelinka I (2006) Investigation on artificial ant using analytic programming. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ACM, New York, NY, USA, GECCO '06, pp 949–950, DOI 10.1145/1143997.1144164. 2006

[Oxford, 2018] Oxford Dictionaries | English. (2018). meme | "Definition of meme in English" by Oxford Dictionaries. [online] Available at: <https://en.oxforddictionaries.com/definition/meme> [Accessed 27 Nov. 2017].

[Ozcan, 2007] Ozcan, E. (2007). "Memes, Self-generation and Nurse Rostering". *Lecture Notes in Computer Science*. Lecture Notes in Computer Science. Springer-Verlag. 3867: 85–104. ISBN 978-3-540-77344-3. doi:10.1007/978-3-540-77345-0_6. 2007.

[Patitz, 2014] Patitz, MJ. (2014) "An introduction to tile-based self-assembly and a survey of recent results." *Natural Computing* 13 (2), 195-224. 2014.

[Picheny et al., 2012] Picheny, V., Wagner, T., Ginsbourger, D. (2012). "A benchmark of kriging-based infill criteria for noisy optimization." Springer. 2012.

[Pintea, 2012] Pintea CM, Chira C, Dumitrescu D, Pop PC (2012) Sensitive ants in solving the generalized vehicle routing problem. *Journal of Computers, Communications and Control*. 6(4):228–231, 1208.5341. 2012.

[Poli, 2007] Poli, R. (2007). "An analysis of publications on particle swarm optimisation applications." En Department of Computer Science, University of Essex, UK. Technical Report CSM-469. 2007.

[Poli, 2007] Poli R, Vanneschi L (2007) Fitness-proportional negative slope coefficient as a hardness measure for genetic algorithms. In: *Proceedings of the 9th Annual Conference on Genetic and evolutionary Computation*, ACM, New York, NY, USA, GECCO '07, pp 1335–1342, DOI 10.1145/1276958.1277209. 2007.

[Prigogine et al., 1996] Prigogine, I. and Stengers, I. (1996). "The end of certainty." New York: Free Press. 1996.

[Prot, 2018] Anon, (2018). [online] Available at: <http://www.proteccion-datos.com.es/2010/los-datos-disociados-y-la-proteccion-de-datos-de-caracter-personal/> [Accessed 1 May 2016].

[Pso, 2018] Anon, (2018). [online] Available at: https://www.researchgate.net/publication/3903911_Particle_swarm_optimization_Development_applications_and_resources [Accessed 3 Feb. 2018].

[Rastrigin, 1974] Rastrigin, L. (1974) "Systems of extremal control." Mir, Moscow. 1974.

[Razmjooy et al., 2016] N. Razmjooy, M. Khalilpour, M. Ramezani. "A New Meta-Heuristic Optimization Algorithm Inspired by FIFA World Cup Competitions: Theory and Its Application in PID Designing for AVR System." *Journal of Control, Automation and Electrical Systems* (2016):1-22, 2016.

[Rego, 2011] Rego C, Gamboa D, Glover F, Osterman C (2011) Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research* 211(3):427–441, DOI 10.1016/j.ejor.2010.09.010

[Ridao et al., 1998] Ridao, M.; Riquelme, J.; Camacho, E.; Toro, M. (1998). "An evolutionary and local search algorithm for planning two manipulators motion". *Lecture Notes in Computer Science*. Lecture Notes in Computer Science. Springer-Verlag. 1416: 105–114. ISBN 3-540-64574-8. doi:10.1007/3-540-64574-8_396. 1998.

[Robo, 2018] Robowiki.net. (2018). "Grammatical Swarm Evolution" - RoboWiki. [online] Available at: http://robowiki.net/wiki/Grammatical_Swarm_Evolution [Accessed 15 Nov. 2017].

[Rodgers, 1985] Rodgers, D.P. (1985). "Improvements in multiprocessor system design". *ACM SIGARCH Computer Architecture News*. New York, NY, USA: ACM. 13 (3): 225–231. doi:10.1145/327070.327215. ISBN 0-8186-0634-7. ISSN 0163-5964. 1985.

[Rosenbrock, 1960] Rosenbrock, H.H. (1960). "An automatic method for finding the greatest or least value of a function". *The Computer Journal*. 3 (3): 175–184. doi:10.1093/comjnl/3.3.175. ISSN 0010-4620. 1960.

[Rothlauf, 2005] Rothlauf F, Oetzel M (2005) On the locality of grammatical evolution. Working Paper 11/2005, Department of Business Administration and Information Systems, University of Mannheim, D-68131 Mannheim, Germany. 2005.

[Rozenberg et al., 2012] Rozenberg, G, Bäck, T., Kok, J., Editors, "Handbook of Natural Computing", Springer Verlag, 2012.

[Ryan et al., 1998] Ryan, C. O'Neill, M. (1998) "Grammatical Evolution: Evolving Programs for an Arbitrary Language". Springer Berlin Heidelberg, Berlin, Heidelberg, pp 83–96. DOI 10.1007/BFb0055930. 1998.

[Saremi et al., 2017] S. Saremi, S. Mirjalili, A. Lewis, "Grasshopper Optimisation Algorithm: Theory and Application". *Advances in Engineering Software* 105, pp 30-47. 2017.

[Schwefel, 1995] Schwefel H. P. (1995). "Evolution and optimum seeking". Wiley New York. 1995.

[Scribd, 2018] Scribd. (2018). "Discrete PI and PID Controller Design and Analysis for Digital Implementation." [online] Available at: <https://www.scribd.com/doc/19070283/Discrete-PI-and-PID-Controller-Design-and-Analysis-for-Digital-Implementation> [Accessed 17 Feb. 2016].

[Sedgewick, 1998] Sedgewick, R. (1998). "Algorithms In C: Fundamentals, Data Structures, Sorting, Searching", Parts 1-4(3 ed.). Pearson Education. 1998.

[Seyedali et al., 2014] Seyedali M., Mirjalili S., and Lewis, A. "Grey wolf optimizer." *Advances in Engineering Software* 69:46-61, 2014.

[Sfu, 2018] Sfu.ca. (2018). "Optimization Test Functions and Datasets". [online] Available at: <https://www.sfu.ca/~ssurjano/optimization.html> [Accessed 2 Aug. 2017].

[Shareef et al., 2015] H. Shareef, A.A. Ibrahim, A.H. Mutlag. "Lightning search algorithm". *Applied Soft Computing* 36(, pp.)1):315-333, 2015.

[Shi, 1998] Shi, Y. (1998). "Parameter selection in particle swarm optimization". *Proceedings of Evolutionary Programming VII (EP98)*. 1998.

[Shi et al., 1998] Shi, Y., Eberhart, R. (1998). "A modified particle swarm optimizer". *Proceedings of IEEE International Conference on Evolutionary Computation*. 1998.

[Shi, 2007] Shi XH, Liang YC, Lee HP, Lu C, Wang QX (2007) Particle swarm optimization-based algorithms for tsp and generalized tsp. *Inf Process Lett* 103(5):169–176, DOI 10.1016/j.ipl.2007.03.010. 2007

[Si et al., 2018] Si, T., De, A. and Bhattacharjee, A. (2018). "Grammatical Swarm Based-Adaptable Velocity Update Equations in Particle Swarm Optimizer." [online] Academia.edu. Available at: http://www.academia.edu/7420024/Grammatical_Swarm_Based-Adaptable_Velocity_Update_Equations_in_Particle_Swarm_Optimizer [Accessed 3 Oct. 2016].

[Simionescu et al., 2002] Simionescu, P.A., Beale, D. (2002). "New Concepts in Graphic Visualization of Objective Functions." *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Canada. p. 891–897. 2002.

[Smith, 2007] Smith J. E. (2007). "Coevolving Memetic Algorithms: A Review and Progress Report". *IEEE Transactions on Systems Man and Cybernetics - Part B*. 37 (1): 6–17. doi:10.1109/TSMCB.2006.883273. 2007.

[Stefan, 2002] Stefan, G. "Chaotic Membrane Computation with Cellular Automata". Manuscript Publishing. 2002.

[Stewart et al., 2015] Stewart, G. L.; Manges, K. A.; Ward, M. M. (2015). "Empowering sustained patient safety: the benefits of combining top-down and bottom-up approaches". *Journal of nursing care quality*. 30 (3): 240–246. doi:10.1097/ncq.000000000000103. 2015.

[Stutzle, 2000] Stützle T, Hoos HH (2000) Max-min ant system. *Future Gener. Comput. Syst.* 16(9):889–914. 2000.

[Stutzle, 1999] Stutzle T, Dorigo M (1999) ACO Algorithms for the Traveling Salesman Problem. *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*, Willey. 1999.

[Swarm, 2018] Swarmintelligence.org. (2018). "Particle Swarm Optimization." [online] Available at: <http://www.swarmintelligence.org/> [Accessed 3 Feb. 2018].

[Taherdangkoo et al., 2012] Taherdangkoo, M; Pazires, M; Yazdi, Mehran; Bagheri, Mohammad Hadi (19 November 2012). "An efficient algorithm for function optimization: modified stem cells algorithm." *Central European Journal of Engineering* 3 (1): 36–50. doi:10.2478/s13531-012-0047-8. 2012.

[Tan et al., 2007] Y. Tan, Z.M. Xiao. "Clonal particle swarm optimization and its applications", in *IEEE Congress on Evolutionary Computation, CEC 2007*, (2007), pp. 2303–2309. 2007.

[Tan et al., 2010] Y. Tan and Y. Zhu (2010). "Fireworks algorithm for optimization." *ICSI 2010, Part I, LNCS 6145*, pp. 355-364. 2010.

[Thomas, 1995] Bäck, Thomas (1995). "Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms." Oxford: Oxford University Press. p. 328. ISBN 0-19-509971-0. 1995.

[Ting, 2005] Ting, C. (2005). "On the Mean Convergence Time of Multi-parent Genetic Algorithms Without Selection." *Advances in Artificial Life*: 403–412. ISBN 978-3-540-28848-0. 2005.

[Ullman et al., 1977] Ullman, J., Alfred A. (1977). "Principles of compiler design." Reading, Mass: Addison-Wesley Pub. Co. pp. 471–2. ISBN 0-201-10073-8. 1977.

[Urbano et al., 2018] Paulo Urbano and Loukas Georgiou, (2018). "Improving Grammatical Evolution in Santa Fe Trail using Novelty Search" [PDF] Available at: <https://mitpress.mit.edu/sites/default/files/titles/content/ecal13/978-0-262-31709-2-ch137.pdf> [Accessed 11 Sep. 2018].

[Wang, 2009] Wang Z, Geng X, Shao Z. (2009) An effective simulated annealing algorithm for solving the traveling salesman problem. *Journal of Computational and Theoretical Nanoscience* 6(7):1680–1686, DOI 10.1166/jctn.2009.1230

[Wadsworth, 1960] Wadsworth, G. P. (1960). “Introduction to Probability and Random Variables”. New York: McGraw-Hill. p. 52. 1960.

[Wilson, 1975] Wilson, E. (1975). “Sociobiology: The new synthesis.” Cambridge, MA: Belknap Press. 1975.

[Wolpert et al., 1997] Wolpert, D.H., Macready, W.G. (1997), “No Free Lunch Theorems for Optimization”, *IEEE Transactions on Evolutionary Computation* 1, p. 67-82. 1997.

[Wong, 2009] Wong KY, See PC (2009) A new minimum pheromone threshold strategy (mpts) for max-min ant system. *Appl Soft Comput* 9(3):882–888, DOI 10.1016/j.asoc.2008.11.011. 2009.

[Xiang, 2015] Xiang W, Yin J, Lim G (2015) An ant colony optimization approach for solving an operating room surgery scheduling problem. *Comput Ind Eng* 85(C):335–345, DOI 10.1016/j.cie.2015.04.010

[Xiao, 2011] Xiao J, Li L (2011) A hybrid ant colony optimization for continuous domains. *Expert Syst Appl* 38(9):11,072–11,077, DOI 10.1016/j.eswa.2011.02.151

[Xue et al., 2016] Xue, J. Wang, Y. Li, H. Meng, X. Xiao, J. “Advanced Fireworks Algorithm and Its Application Research in PID Parameters Tuning.” Air Force Engineering University. 2016.

[Yang, 2008] Yang, X. “Nature-inspired metaheuristic algorithms”, Luniver Press, 2008.

[Yang, 2016] Yang, X. Deb, S. Fong, S. He, X. Zhao, YX. “From Swarm Intelligence to metaheuristics: Nature-inspired optimization algorithms”, *Computer* 49 (9): 52-59. 2016

[Zheng et al., 2017] Zheng, S., Li, J., Janecek, A. and Tan, Y. (2017). “A Cooperative Framework for Fireworks Algorithm.” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(1), pp.2-41. 2017.

ANEXOS

Anexo I: Tabla de Objetivos contra Resultados de la Tesis Doctoral

A continuación se presentan los objetivos, de índole teórico-práctica, expuestos en el epígrafe *1.1 Objetivos*, junto con su descripción y estado de completitud:

Objetivo	Descripción	Estado	Observación
[OBJ-1]	Ofrecer una exhaustiva visión panorámica del estado del arte del Paradigma de la Computación Natural, constituyendo una base teórica para la Tesis Doctoral. Durante dicho proceso, se introducirá la problemática global y su perspectiva. Durante su desarrollo, se velará por la introducción de los posibles nuevos paradigmas adscritos al espectro de la Computación Natural y Bioinspirada que puedan ver la luz, manteniendo un estado del arte de rabiosa actualidad. Asimismo, en los casos oportunos, se definirá un repositorio software de ejemplos de aplicación para cada metodología algorítmica adscrita al paradigma objeto de estudio.	✓	Especialmente en el Capítulo 2. <i>Estado del Arte</i> . Repositorios implementados y disponibles en https://github.com/ClemencioMoralesLucas/PhD & https://github.com/ClemencioMoralesLucas/TensorFlow
[OBJ-2]	Alcanzar una sinergia entre Ant Colony Optimization (ACO) y Grammatical Evolution (GE). Emplear dicha sinergia partiendo de la hipótesis de usar ACO para explorar una gramática BNF cuyos elementos son la solución a un problema dado. Implementar la sinergia entre ACO y GE a nivel de software.	✓	Sinergia conseguida, dando lugar al nacimiento del novedoso algoritmo ACORD (Ant Colony systems Optimization applied to BNF grammars Rule Derivation).
[OBJ-3]	Mejorar empíricamente (en términos de rendimiento) Fireworks Algorithm (FA), partiendo de la hipótesis de que podría variarse a nivel conceptual para lograr un mejor desempeño. Implementar los algoritmos resultantes a nivel de software.	✓	Mejora conseguida, dando lugar a la concepción de IFA (Improved Fireworks Algorithm).

Tabla X: Tabla de Objetivos contra Resultados de la Tesis Doctoral

Anexo II: Publicaciones relacionadas con la Tesis Doctoral

Seguidamente se introducen las distintas y más reseñables publicaciones, ponencias, lecturas en congresos y contribuciones, en términos generales, relacionados con la presente Tesis Doctoral.

En primer lugar, las distintas publicaciones y conferencias impartidas se enumeran y detallan a continuación:

- Publicación del artículo JCR Q2 “*Natural Computing Applied to the Underground System: A Synergistic Approach for Smart Cities*” en la revista *Sensors*. [Morales Lucas et al, 2018].
- Publicación del artículo JCR Q2 “*Ant Colony systems Optimization applied to BNF grammars Rule Derivation (ACORD Algorithm)*” en la revista *Soft Computing*. [de Mingo et al, 2019].
- Creación del artículo (en proceso de presentación para JCR Q2) “*Pattern Recognition with Convolutional Neural Networks: Humpback Whale Tails*”.
- Impartición de conferencia sobre el artículo “*Collective Computation: Turning the Underground into an Ant Nest*” en el *Institute of Information Theories And Applications - 2016 Edition* (ITHEA - <http://www.ithea.org/institute/institute.html>) Varna. (Bulgaria).
- Impartición de conferencia sobre el artículo “*Applying Natural Computing to the Urban Public Transport Systems: A pragmatic approach*” en el *Workshop on Computing Science and Technology for Smart Cities 2016* (COSTAC 2016) - Madrid. (España).
- Impartición de conferencia sobre el artículo “*The Natural Computation Paradigm: Bio-inspired Algorithms for Smart Cities*” en el *Workshop for Young Researchers in Applied Mathematics and Engineering 2016*, Madrid. (España).
- Impartición de conferencia sobre el artículo “*Fireworks Algorithm Applied to Smart Cities field & Optimizations*” en el *Research Seminar of the Technical University of Madrid, 2017, Master in Computing Science and Technologies*. Universidad Politécnica de Madrid, Madrid. (España).

- Impartición de conferencia sobre el artículo "*Fireworks Algorithm: A Pragmatic Approach Based On Gaussian Explosion for Sparks*" en el *Workshop on Computing Science and Technology for Smart Cities 2017* (COSTAC 2017) - Madrid. (España).

Por otro lado, figura seguidamente la asistencia a congresos, conferencias, cursos, ponencias y todo tipo de eventos relacionados con la investigación y el ámbito de la Tesis Doctoral:

- Asistencia a numerosas conferencias y seminarios de interés (a nivel europeo) en el ámbito de la investigación, no sólo en el ámbito de la Computación Natural, sino de la Ingeniería Informática en general. Es destacable que se ha acudido a un número mayor de seminarios/ponencias/conferencias/ciclos de las requeridas por el Programa de Doctorado en Ciencias y Tecnologías de la Computación para Smart Cities. Entre ellas, destacan conferencias en prestigiosas instituciones universitarias, como:
 - Universidad Politécnica de Madrid (<http://www.upm.es/>): Numerosos eventos relacionados con la investigación desde incluso antes de comenzar oficialmente el Programa de Doctorado en Ciencias y Tecnologías de la Computación para Smart Cities.
 - Universidad Complutense de Madrid (<https://www.ucm.es/>): En la misma línea, asistencia a numerosos eventos de importancia relacionados con la investigación científica.
 - Massachusetts Institute of Technology (MIT) Open Courses (<http://web.mit.edu>): Desde el año 2012, realizo con frecuencia cursos de <https://ocw.mit.edu/index.htm>, especialmente aquellos relacionados con los temas principales de investigación de la Tesis Doctoral. Dichos cursos se extraen de asignaturas reales del MIT, impartidos por la entidad estadounidense de forma abierta, gracias a los cuales he podido extraer numerosa información de utilidad que ha sido aplicada a esta Tesis Doctoral.
 - Imperial College London. (<https://www.imperial.ac.uk/>): Destacan principalmente "Open Sessions" caracterizadas por la exposición de trabajos de investigación sobre Computer Science & Engineering:
 - Open Sessions - Imperial College Computational Bioinformatics Laboratory (CBL) - (2015).

- Varios cursos abiertos y eventos relacionados con los campos Computer Science/Computer Engineering en particular y Natural Computing en particular, fundamentales en la Tesis Doctoral que nos ocupa.
- University College London. (<https://www.ucl.ac.uk>): Destacan los siguientes cursos y eventos, entre otros:
 - Robotics and Autonomous Systems work showcased - Prof Mandayam Srinivasan. (Enero 2015) http://www.cs.ucl.ac.uk/archive/article/robotics_and_autonomous_systems_work_showcased/. Es destacable que en este evento pude, además de adquirir conocimientos e ideas que utilicé posteriormente en mi Tesis Doctoral, tomar conciencia del programa Horizon 2020, clave en el futuro de las Smart Cities, a su vez muy presentes en el Programa de Doctorado en Ciencias y Tecnologías de la Computación para Smart Cities. El programa Horizon 2020 y las Smart Cities poseen un anexo dedicado en la presente Tesis Doctoral, conteniendo información que, en gran mayoría, proviene de este evento y de las fuentes que en el mismo obtuve. Por otro lado, también disfruté de la oportunidad de adquirir valioso conocimiento técnico por parte de profesores del prestigioso MIT de Massachusetts en dicho evento.
 - 2nd International Conference on Urban Sustainability & Resilience: (Noviembre, 2014) <http://www.ucl.ac.uk/usar/pdf/usar2014conferenceprogramme> En el presente evento pude reforzar mis conocimientos sobre Smart Cities, adquirir ideas para posteriores investigaciones relacionadas con la presente Tesis Doctoral, estudiar diversas publicaciones al respecto de las Smart Cities y valorar enfoques de Data Mining para Smart Cities. Dado el amplio espectro que cubren las conferencias del evento, destacaría especialmente dos de las mismas, especialmente interesantes y de las que pude extraer numerosa información para mi Tesis Doctoral:
 - 1) Resilient Urban Systems & Networks (Interdependence and Change) - Victoria Sword.
 - 2) A Data Mining Approach to Seismic Fragility Analysis of Road and Rail Bridges - Khaled Paul Taalab, Pierre Gehl, Tao Cheng, Dina D'Ayala.

- Por otro lado, también se ha asistido a prestigiosas conferencias internacionales, como la multitudinaria *London Computing Conference 2017* (<http://saiconference.com/Conferences/Computing2017>).
- Asistencia al "*Seminario de Investigación 2017*" (Marzo-Mayo, 2017) de la Escuela Técnica Superior de Ingeniería de Sistemas Informáticos (ETSI - <https://www.etsisi.upm.es>) de la Universidad Politécnica de Madrid (UPM), adscrito al Programa de Doctorado en Ciencias y Tecnologías de la Computación para Smart Cities (<https://www.etsisi.upm.es/estudios/doctorado/61d1/pa/SemInv>).
- Asistencia al "*Seminario de Investigación 2016*" (Febrero-Abril, 2016) de la Escuela Técnica Superior de Ingeniería de Sistemas Informáticos (ETSI - <https://www.etsisi.upm.es>) de la Universidad Politécnica de Madrid (UPM), adscrito al Programa de Doctorado en Ciencias y Tecnologías de la Computación para Smart Cities (<https://www.etsisi.upm.es/estudios/doctorado/61d1/pa/SemInv>).
- Asistencia al congreso ISCRAM-med 2016 (<http://dei.inf.uc3m.es/isqrammed2016/>) Madrid. (España).
- Asistencia al curso "*Preparación para la escritura de artículos científicos y para la exposición de resultados de investigación*", impartido por Prof. Dr. José Eugenio Naranjo Hernández en la Universidad Politécnica de Madrid (U.P.M) y redacción del trabajo final asociado a dicho curso.
- Asistencia al Congreso "*JBCN Conference 2017*" en Barcelona (España) donde se acudió a más de 12 conferencias en los 3 días que duró el evento. Es reseñable que, gracias a la asistencia al evento, numerosas técnicas de optimización (especialmente *software*) han podido ser aplicadas a la presente Tesis Doctoral.
- Persiguiendo el mismo objetivo que en "*JBCN Conference 2017*", asistencia al Congreso Codemotion 2015 en Madrid (España).
- Asistencia al Congreso Codemotion 2016 en Madrid (España) con el mismo fin.
- De manera análoga, asistencia al Congreso Codemotion 2017 en Madrid (España).

Finalmente, se presenta en la tabla el grado de cumplimiento de todos los objetivos relacionados con el anexo actual, que figuran en la normativa del Programa de Doctorado en Ciencias y Tecnologías de la Computación para Smart Cities:

Objetivo	Descripción	Estado	Observación
[OBJ-1]	Se ha publicado un artículo en una publicación adscrita a JCR (<i>Journal Citation Reports</i>)	✓	Se ha superado el número, publicando dos artículos JCR en vez de uno: [Morales Lucas et al, 2018] y [de Mingo et al, 2019].
[OBJ-2]	Se han impartido las conferencias anuales por curso y año contemplados en el Programa de Doctorado.	✓	De nuevo se ha superado el número, impartiendo más de las requeridas.
[OBJ-3]	Se ha asistido al número de congresos/ponencias/workshops anuales por curso y año contemplado en el Programa de Doctorado.	✓	Una vez más se ha superado el número, impartiendo más de las requeridas.

Tabla XI: Tabla de Objetivos del Programa de Doctorado en Ciencias y Tecnologías de la Computación

Anexo III: Líneas Prácticas de Investigación y Aplicación a Smart Cities

A.1 Líneas Prácticas

En lo referido a las líneas prácticas de ampliación de la presente Tesis Doctoral, es destacable que estas están centradas, principalmente, en el ámbito de las Smart Cities. (Tal y como el Programa de Doctorado en cuestión, situado en el marco de las Ciencias de la Computación y las Smart Cities.) En ellas, se planteará cómo puede aplicarse el paradigma de la Computación Natural a dichos núcleos urbanos, que serán definidos. Finalmente, se relatarán los ejemplos más sobresalientes de Smart Cities tanto a nivel nacional como internacional, así como la justificación de las mismas, donde se analizarán alternativas de resolución existentes y se presentarán nuevos esquemas a modo de vías de ampliación de la presente Tesis Doctoral.

Los sucesivos epígrafes tienen el objetivo de describir, de forma pormenorizada, las principales vertientes prácticas más notables, fruto del presente trabajo de investigación.

A.1.1 Sistema Enjambre

Persiguiendo la consecución de una poderosa sinergia entre varios conceptos adscritos a la Computación en Enjambre y Natural, innegablemente atractiva para el entorno de las smart cities, se propone la creación de un sistema que permita el estudio, en tiempo real, del comportamiento a grandes rasgos de los ciudadanos que utilizan el transporte público subterráneo de una smart city. Dicho sistema ha sido publicado en [Morales Lucas et al, 2018].

De forma atómica, la meta del sistema consiste en, apoyándose en los *smartphones* que un gran porcentaje de la población posee, estudiar el comportamiento de dicho grupo muestral mediante el uso de datos disociados como población estadística. El sistema propuesto con anterioridad haría posible, entre otros factores, los relatados seguidamente:

- Conocer de primera mano las rutas más frecuentadas en función del uso por parte de la ciudadanía, así como el comportamiento en cuanto a rutas de los usuarios en el transporte público. Dicho factor constituye una puerta abierta a una eficiente gestión en pos de la optimización del sistema.
- Preparar, en base a las estimaciones obtenidas mediante el estudio de los datos recogidos, el sistema de transporte público para lidiar con máximos

absolutos de carga. Tal escenario puede obtenerse estudiando los valores atípicos en cuanto al uso de las instalaciones.

- Detectar emergencias, tales como un tren bloqueado en medio de un túnel o casuísticas en las que el número de personas en el andén es excesivo, superando el aforo de la dársena en cuestión, por citar algunos ejemplos.
- Preparar rutas alternativas en caso de uso intensivo y/o averías para los pasajeros.
- Ampliarse en cuanto a la inclusión de nuevas funcionalidades actualmente en evaluación y delimitación.

Con el fin de llevar a cabo la captación de datos, se propone el despliegue de una serie de balizas diseminadas estratégicamente por la red de metro, que enviarán la información captada a un servidor de datos para realizar el estudio de la información obtenida. La siguiente figura tiene como objetivo el esclarecimiento de la topología y el comportamiento de dicha red de balizas, que compondrá una Wireless Sensor Network (WSN):

Figura LXXXIV: Fotografía esquemática del sistema de componentes de la Wireless Sensor Network

Tal y como puede apreciarse en la figura superior, la presente subsección del sistema se escinde en los elementos A, B y C, a saber:

- A: Baliza fija: En terminología de WSN, este microcomputador *smart dust* (también denominado mota), que será único en cada estación, estará integrado en la dársena. En base a un esquema maestro-servidor, la baliza fija interpretará el rol de maestro. La baliza fija recibirá información de las balizas móviles, y se comunicará únicamente con el Centro de Control y Operaciones. Ambos elementos se definirán seguidamente.
- B: Baliza móvil: El presente tipo de mota, cuyo número podrá ser superior a uno en función de su rango de alcance, interpretará el rol de esclavo. Las balizas móviles interpretarán su rol en el sistema integradas en el propio vehículo del STPU (Sistema de Transporte Público Urbano), y se comunicarán, de forma singular, con la baliza fija de cada estación.
- C: Terminal de Usuario (TU): Smartphone de los usuarios del STPU, que será utilizado para evidenciar su presencia en la estación, así como para las comunicaciones pertinentes. Los terminales de usuario gozarán de la potestad de comunicarse tanto con las balizas fijas como con las balizas móviles, siendo esta comunicación necesaria para informar de su presencia en la estación y su ingreso y egreso en el tren, respectivamente.

A continuación figura un diseño esquemático de la posición de los elementos A,B y C:

Figura LXXXV: Diseño esquemático del sistema enjambre propuesto

Una vez definida la topología de la WSN, formada por la baliza fija de cada dársena y las balizas móviles equipadas en el tren adscrito al STPU, se procederá a la descripción del Centro de Control y Operaciones (CCO), cuya apariencia esquemática se evidencia en la figura anexa:

Figura LXXXVI: Imagen ejemplificativa del Centro de Control y Operaciones

Como se observa en la figura, el CCO se verá constituido por la siguiente dupla de elementos:

- D: Terminal de Control: Recibe los datos enviados por las balizas fijas de cada estación, mostrando el estado pertinente y la existencia, si procede, de situaciones anómalas desde el punto de vista de seguridad y/o fallo en los sistemas.
- E: Administrador del Sistema: Evalúa los datos mostrados y opera en consecuencia al estado actual de los STPU y su entorno, bien sea disparando la ejecución de un determinado protocolo de seguridad ante fallos si procede o resolviendo las distintas situaciones anormales que podrían presentarse.

A.1.2 Aplicación basada en Híbrido FA-ACO para zonas rurales

Tal y como se avanza en la sección circunscrita (en este mismo capítulo) a la propuesta teórica híbrida entre FA-ACO, esta vertiente presenta una aplicación práctica de directo impacto en el ámbito de las smart cities, que será descrita a continuación.

Supóngase una zona boscosa anexa a una montaña, en la que existe un punto de salvamento. Si tal centro se considerara el punto donde FA lanza el firework,

las *sparks* bien podrían ser drones equipados con cámaras, que avanzan desde el centro de control o salvamento hacia la ubicación determinada, como si de *sparks* se trataran.

Durante dicho camino, la cámara instalada en cada dron podría escanear el terreno. Es, precisamente en este momento, donde entra en juego una de las bibliotecas más prometedoras de los últimos años; Google Tensor Flow (GTW en lo sucesivo).

De manera atómica, GTW constituye una API de detección de objetos a través de imágenes, basado en un motor de *machine-learning*. La biblioteca goza de una capacidad casi únicamente acotada por la imaginación, siendo su potencia directamente proporcional a su precisión. El siguiente repositorio, preparado en el ámbito de la presente Tesis Doctoral y que será a continuación detallado, da buena cuenta de la efectividad de Tensor Flow: <https://github.com/ClemencioMoralesLucas/TensorFlow>

Supóngase la siguiente imagen como entrada del software implementado en el repositorio anterior:

Figura LXXXVII: *Puppy* - Imagen de entrada para el repositorio *TensorFlow*

A grandes rasgos, el código fuente tiene la misión de evaluar, a nivel binario, la imagen que recibe por parámetro, realizando un contraste, con diferentes gradientes, contra su base de datos de imágenes conocidas, en las que ha sido entrenado. La sección de código que realiza a bajo nivel tal comparación se muestra a continuación, quedando el lector referenciado al repositorio *per se* para un mayor nivel de detalle:

```
private static Tensor<Float> constructAndExecuteGraphToNormalizeImage(byte[] imageBytes) {
    try {
        Graph g = new Graph();
        GraphBuilder b = new GraphBuilder(g);

        final int H = 224;
        final int W = 224;
        final float mean = 117f;
        final float scale = 1f;

        final Output<String> input = b.constant("input", imageBytes);
        final Output<Float> output =
            b.div(
                b.sub(
                    b.resizeBilinear(
                        b.expandDims(
                            b.cast(b.decodePeg(input, 3), Float.class),
                            b.constant("make_batch", 0)),
                        b.constant("size", new int[]{H, W}),
                        b.constant("mean", mean)),
                    b.constant("scale", scale));
        try {
            Session s = new Session(g);
            return s.runner().fetch(output.op().name()).run().get(0).expect(Float.class);
        }
    }
}

private static float[] executeInceptionGraph(byte[] graphDef, Tensor<Float> image) {
    try {
        Graph g = new Graph();
        g.importGraphDef(graphDef);
        try {
            Session s = new Session(g);
            Tensor<Float> result =
                s.runner().feed("input", image).fetch("output").run().get(0).expect(Float.class);
            final long[] rshape = result.shape();
            if (result.numDimensions() != 2 || rshape[0] != 1) {
                throw new RuntimeException(
                    String.format(
                        "Expected model to produce a [1 N] shaped tensor where N is the number of labels, instead it produced one with shape '%s'",
                        Arrays.toString(rshape)));
            }
            int nlabels = (int) rshape[1];
            return result.copyTo(new float[nlabels])[0];
        }
    }
}
```

Figura LXXXVIII: Métodos de *mapping*, reducción y normalización en el código fuente del ejemplo TensorFlow

Al ejecutar el *software*, se obtiene el siguiente *output*, representativo del nivel de éxito de la biblioteca en su evaluación de la imagen recibida:

Figura LXXXIX: Resultado del software al analizar la imagen de *Puppy (Canis Lupus Familiaris)*

Por consiguiente, el *software* detecta que el animal es un *dingo* (raza de can salvaje autóctona de Australia), con un 38,51% de posibilidades. Adviértase que este porcentaje presenta una gran precisión dada la baja calidad de la imagen, tomada por un terminal móvil telefónico de 2007.

Es lícito, *a posteriori*, probar la efectividad del *software* con una imagen de mayor calidad. Adviértase que dicha imagen constituye todo un reto para un software de análisis gráfico, puesto que existen numerosos reflejos en un entorno que, además, es acuático:

Figura XC: Gran Tiburón Blanco - Imagen de entrada para el repositorio TensorFlow

Tras lanzar el programa, se obtiene el siguiente *output*, que demuestra la apabullante efectividad de la aplicación frente a imágenes de alta calidad:

```
LabelImage
/usr/lib/jvm/java-8-openjdk-amd64/bin/java ...
2018-07-01 10:42:21.924246: I tensorflow/core/p
BEST MATCH: great white shark (98.54% likely)
Process finished with exit code 0
```

Figura XCI: Resultado del software al analizar el Gran Tiburón Blanco (Carcharodon Carcharias)

Es decir, el programa determina que el animal de la fotografía es, con un 98.54% de probabilidad, un Gran Tiburón Blanco. Es reseñable que la calidad extrema de los *appliances* fotográficos de 2018, permitirían una tasa de reconocimiento efectiva cercana a la obtenida en la imagen anteriormente mostrada.

Tales ejemplos bien podrían extrapolarse, retomando el ejemplo que abre el presente epígrafe, a los drones que peinan la zona boscosa, que podrían ser dotados de cámaras y un software como el anteriormente mostrado. Dicha sinergia permitiría la obtención de imágenes como la siguiente, donde los objetos individuales son analizados y descritos, así como la tasa de éxito de su evaluación:

Figura XCII: Imagen tipo, susceptible de constituir el *output* de la flota de drones

Los conceptos mostrados en el presente epígrafe desembocan en un interesante sistema de rescate y salvamento: El centro de control de la zona boscosa, que constituirá el punto de lanzamiento del *firework*, lanzará a su vez los drones equipados con cámaras y el software pertinente hacia una zona concreta, actuando a modo de *sparks*. En caso de detectar a un posible montañero perdido en la zona, reconociendo la figura inmediatamente, lanzará una señal de aviso, que contiene la localización del montañero, al centro de control. Un esbozo esquemático de dicho sistema puede apreciarse en la siguiente figura:

Figura XCIII: Diseño esquemático del sistema de rescate basado en FA y la flota de drones

Es, precisamente, en el momento en el cual un dron detecta a un montañero cuando ACO entra en acción: Si los drones *trazan* su itinerario, enviándolo al centro de control, se estaría en disposición de hallar la ruta mínima desde el montañero al centro de control y viceversa mediante la aplicación del algoritmo ACO. De esta forma, se lograría la deseada característica de estigmergia que caracteriza a dicha técnica, dotando a los drones de una suerte de inteligencia colectiva, en cierto modo, que permitiría detectar a personas perdidas en entornos tan hostiles como el montañoso.

A.2 Aplicación a Smart Cities

Una vez descritas pormenorizadamente las líneas prácticas anteriores, de manera obvia orientadas a smart cities, el presente sub-capítulo consta de la misión de definir a qué se corresponde este término, de popularidad tan alta como ascendente en el mundo IT, citando ejemplos de las mismas así como su justificación, entre otros factores reseñables.

A.2.1 Smart Cities

Se denomina “Smart city” o Ciudad Inteligente a aquel entorno urbano capaz de autogestionarse, en cuanto a términos energéticos y de control se refiere. Tal concepto ha sido introducido como un mecanismo estratégico para abarcar modernos factores de producción urbanos en un marco común y para poner de relieve, a su vez, la creciente importancia de las Tecnologías de la Información y la Comunicación (también conocidas por su acrónimo, TIC).

En dicho ámbito, son las WSN (*Wireless Sensor Networks*; Redes de Sensores Inalámbricas) quienes constituyen una tecnología vital para la creación y el desarrollo de ciudades inteligentes, permitiendo concebir una macro-red distribuida de sensores que tendrán como objetivo la medición de todos y cada uno de los parámetros que permitan llevar a cabo la gestión de la ciudad de la manera más eficiente posible. Este conjunto de parámetros, tales como los que ilustra la siguiente figura, serán enviados de forma inalámbrica y en tiempo real a los ciudadanos y/o a las autoridades pertinentes según sea oportuno:

Figura XCIV: Datos susceptibles de monitorización en una smart city

Adviértase que de esta forma se posibilita el control de elementos tan fundamentales como la concentración de la contaminación ambiental en toda la ciudad, el estado global de los STPU así como la optimización del riego de zonas verdes. Por otro lado, elementos tales como el flujo de vehículos a lo largo de la ciudad puede ser monitorizado en pos de gestionar de forma eficiente los sistemas de iluminación de la misma, que pueden ser deshabilitados si el número de vehículos en cierta zona es suficiente como para que sus luces de posición hagan posible la visión. De manera paralela, el tráfico puede ser minimizado

gracias a las WSN, que podrían indicar a los conductores que lo deseen la plaza de aparcamiento más cercana, lo que a su vez se traduce en un ahorro de tiempo y combustible, así como en la disminución de los atascos y en la mejora de la calidad de vida de la ciudad. Por otro lado, puede ser factible, incluso, la instalación de sensores en los contenedores de basura, que envíen una señal de aviso a los servicios de limpieza cuando estén cerca de llenarse. Gracias a dicho comportamiento, el protocolo de recogida de basura puede optimizarse en base a la aplicación de numerosas metodologías adscritas a la Teoría de Grafos.

Tal y como se relatará en sucesivos epígrafes del presente capítulo, es digno de mención que, tal como Ámsterdam [Amsterdam, 2018] y Lyon [Lyon, 2018] consiguieron para sus respectivos países, Málaga [Malaga, 2018] ha hecho del término *smart city* una prácticamente total realidad en España. Además, el proyecto de dicha ciudad andaluza pretende conseguir una integración óptima de las fuentes renovables de energía en la red eléctrica, acercando las curvas de oferta y demanda a través del establecimiento de nuevos modelos de gestión basados en la micro generación eléctrica. Se pretenden gestionar, por otro lado, los sistemas de almacenamiento energético en baterías para facilitar su consumo posterior en la climatización de edificios, el alumbrado público y el transporte eléctrico, a la vez que se persigue fomentar el uso de vehículos eléctricos, con la instalación de postes de recarga y la implantación de una flota de dichos medios de transporte.

Para finalizar la implantación de dicho proyecto, se contará con nuevos contadores inteligentes desarrollados en el marco de la telegestión para hacer posible un consumo eléctrico más sostenible. Además, está previsto que la instalación de sistemas avanzados de telecomunicaciones y telecontrol posibiliten una actuación en tiempo real y de forma automática sobre la red de distribución, haciendo posible una nueva gestión de la energía y potenciando la calidad del servicio.

Una vez introducido al lector al rabiosamente actual concepto de las *smart cities*, se realizará un especial hincapié en aquellos aspectos necesarios para dotar a los STPU (Sistemas de Transporte Público Urbano) de la inteligencia propia de una *smart city*. (La justificación de dicho hincapie tiene un epígrafe homónimo en el presente capítulo.) Destaca, en dicha línea, el proyecto de ámbito europeo *City Keys*, que identifica las necesidades actuales de las *smart cities*. Dicho proyecto, a su vez, recoge las necesidades de la ciudadanía y las distintas entidades que en ella residen, con el objetivo de identificar las prioridades y establecer los criterios necesarios que ayuden a la medición del impacto de los proyectos relacionados con las *smart cities* a nivel continental.

El proyecto *City Keys* constituye un marco impulsado por la red *Eurocities* [Eurocities, 2018], que ve la luz en febrero de 2015 con el objetivo de concebir un

sistema de evaluación de resultados. Dicho sistema, idealmente pretende posibilitar la medición y comparación de las soluciones relacionadas con el término *smart city* a nivel europeo, la creación de directrices para una recolección de datos genérica, la concepción de un prototipo de medición probado en distintas urbes y el desarrollo de recomendaciones bidireccionales, tanto a nivel metropolitano como empresarial, que podrían encontrar jugosas y novedosas oportunidades de negocio. La importancia del proyecto *City Keys*, financiado por la Unión Europea en el ámbito del programa *Horizon 2020*, se ve poderosamente respaldada por la entidad de las metrópolis participantes, entre las que destacan Zaragoza, Viena, Zagreb, Rotterdam y Tampere, por citar algunas.

Continuando la línea del interesante proyecto *City Keys*, se destaca la elevada expectativa despertada por el mismo en los organismos municipales, principalmente en las áreas relacionadas con la sostenibilidad, la mejora de la calidad de vida, la participación ciudadana activa y la eficiencia energética. Los anteriormente citados organismos consideran la transición de sus núcleos urbanos a *smart cities* como un punto de inflexión fundamental en la vida ciudadana, así como una vital oportunidad para beneficiarse de importantes ahorros. En efecto, tomando como ejemplo la Comunidad de Madrid y el subconjunto energético del suministro eléctrico, el siguiente ejemplo se antoja hiperbólicamente ilustrativo: Actualmente, la optimización del consumo eléctrico es uno de los principales factores a tener en cuenta a la hora de controlar el impacto económico en cuanto a gastos que genera un núcleo urbano. En 2011, el consumo eléctrico en la Comunidad de Madrid fue del 23,4% sobre el total de gasto energético, valor únicamente superado por el 58,1% del consumo petrolífero. La tendencia al alza de precios en la energía eléctrica, de la mano de la presión medioambiental y socioeconómica, motiva a la sociedad a enfrentar dos variaciones horarias a lo largo del año, persiguiendo la finalidad de aprovechar la luz solar en detrimento de la eléctrica. Ahora bien, a pesar de que esta medida se antoja útil, no es del todo eficiente, puesto que existen determinados sectores en los que, bien debido a su ubicación geográfica, la cercanía de edificios con un potente alumbrado, o una poco precisa gestión, el aprovechamiento lumínico no es óptimo. Baste tomar como premisa el ilustrativo ejemplo que acontece cada año en la citada Comunidad de Madrid, donde, desde que el reloj se atrasa una hora en octubre para ahorrar energía, los Sistemas de Iluminación Urbanos (SIU), tales como las farolas, se encienden, aproximadamente, a las 18:10 horas, cuando la luz natural permite la visión hasta las 18:30. Este intervalo de veinte minutos de gasto eléctrico ineficiente no supone un exacerbado impacto económico si se considera una única farola, pero, teniendo en cuenta que la Comunidad de Madrid posee, en 2015, más de un cuarto de millón de dichos dispositivos, el coste asociado deja de ser en absoluto despreciable, tal y como evidencia el siguiente ejemplo:

Por norma general, el consumo de una farola oscila los 200 W/hora, por lo que se infiere, trivialmente, que dicho gasto es de 67 W cada 20 minutos. Multiplicando dicho valor por el número de días que tiene un mes, el gasto ineficiente se traduce en 620 minutos mensuales. Por tanto, la potencia despilfarrada en el mes de octubre se corresponde con:

$$\begin{aligned}\text{Consumo en octubre} &= \text{Minutos} * W \text{ necesarios} = 620 * 67 = 41540 W \\ &= 41,540 KW\end{aligned}$$

En este punto, puede calcularse el volumen económico que supone dicha potencia extrapolada a las 250.000 farolas que la Comunidad de Madrid posee:

$$\begin{aligned}\text{Cantidad monetaria desperdiciada} &= 41,540 * 0,150938 * 250.000 \\ &= 1.567.491,13 \text{ €}\end{aligned}$$

Por lo que se concluye, a tenor de los datos anteriormente expuestos, que la carencia de automatización en los Sistemas de Iluminación Urbanos se traduce, sólo en el mes de octubre, en un impacto económico superior al millón y medio de euros. Es reseñable recordar que dicho ejemplo únicamente contempla la Comunidad de Madrid y su gasto puramente eléctrico, por lo que la implantación de sistemas de optimización aplicados a esta fuente de energía y sus homólogas se traduciría en un cuantioso ahorro que permitiría gozar a las arcas públicas de una inusitada vitalidad.

Retomando las principales características del proyecto *City Keys*, se destaca que, actualmente, aproximadamente el 60% de las grandes ciudades europeas consultadas afirman tener en marcha algún proyecto ligado a la potenciación de los factores propios de una smart city, principalmente en relación con la movilidad, eficiencia energética y las populares TIC, pues entienden que el concepto smart city está íntimamente ligado al desarrollo tecnológico.

Es digno de mención que la mayoría de ciudades involucradas en el anteriormente citado proyecto coinciden en tres objetivos prioritarios, cuya consecución está directamente relacionada con la consecución de una smart city:

1. El aumento de la calidad de vida de los ciudadanos, proveyendo las herramientas fundamentales para una mejor gestión ciudadana de los quehaceres diarios.
2. La mejora de los servicios existentes en pos de la ciudadanía, entre los que destaca principalmente el Sistema de Transporte Público Urbano (STPU) y su sostenibilidad, entendiendo como tal el respeto por parte de los mismos hacia el medio ambiente.

3. La creación de numerosos puestos de trabajo, necesarios para mantener el estado de los nuevos sistemas de optimización adscritos a la *smart city*.

Cabe destacar, precisamente en relación con la ciudadanía, que este colectivo coincide con los organismos municipales en la benignidad de las smart cities, señalando la tercera necesidad anteriormente propuesta como especialmente prioritaria y deseable. Adviértase que el reto de la creación de empleo supone un interesante reto, que podría solucionar en gran medida la difícil y problemática situación socio-económica actual. Sin embargo, y a pesar de la aparentemente elevada aceptación por parte del concepto de smart city por parte de la ciudadanía, destaca la necesidad de invertir en campañas de concienciación, pues existe aún un alto nivel de desconocimiento hacia dicho término por parte de la mayoría de estratos sociales. En términos cuantitativos, un 25% de los individuos encuestados por el proyecto *City Keys* desconoce el concepto de smart city, siendo incapaz de citar ningún proyecto que pueda considerarse como propio de dicho término. En la misma línea, *City Keys* evidencia la carencia de herramientas que posibiliten la medición de los resultados obtenidos al implementar un sistema ligado al concepto de smart city. Destaca que únicamente el 11% de las ciudades participantes en la encuesta ha llevado a cabo algún tipo de medición a nivel global, mientras que el 61% admite no haber efectuado ningún tipo de medición, ni siquiera en el ámbito local. Por otro lado, el 72% de las áreas urbanas desearía poder llevar a cabo una evaluación integral del impacto de las estrategias implantadas mediante una herramienta, idealmente software, sencilla y carente de incompatibilidades con las ya existentes y en uso. Un esperanzador dato en el ámbito de la filosofía open data, análogo al movimiento open source, es el elevado porcentaje de metrópolis que hacen público sus datos, disponiéndolos al libre uso por parte del ciudadano, siendo dicho porcentaje del 66%.

Se recuerda que uno de los aspectos más importantes en el ámbito de la smart city es la consecución del siguiente conjunto de objetivos, entre otros:

- Consolidar la expansión del núcleo metropolitano, permitiendo una ordenada, controlada y flexible evolución.
- Obtener una visión panorámica e integrada de absolutamente todos los ámbitos de la metrópolis, de tal forma que se posibilite la existencia de sinergias y ahorros tan significativos como los especificados en cuanto a la energía eléctrica en el presente epígrafe.
- Otorgar a la ciudadanía un óptimo conjunto de servicios, de calidad suprema y menor coste, factores que convergen en una gestión sostenible.

Aunando varios de los conceptos descritos a lo largo del epígrafe actual, el siguiente Diagrama de Venn refleja fielmente los compromisos que pretende respetar toda smart city que se precie:

Figura XCV: Diagrama de Venn ejemplificativo de los compromisos de una *smart city*

De forma paralela, no debe olvidarse un importante factor dentro del ámbito de la smart city, y es que la elevada cantidad de sistemas de control adscritos a la misma generarán ingentes cantidades de datos, del orden del *terabyte*. Por tal motivo, frente a la necesidad imperiosa de gestionar tan enorme orden de datos, el Paradigma de Computación Big Data constituye un aliado fundamental en el control de la smart city. Este emergente paradigma computacional se corresponde con el elemento aglutinante en el ámbito del control de datos de la smart city, colaborando de manera inestimable a una óptima gestión de los recursos inteligentes de la metrópolis.

Es importante reseñar nuevamente, en el ámbito de las smart cities, que los STPU constituyen uno de los focos prioritarios de mejora. Ahora bien, ¿cuáles son los principales problemas de los STPU contemporáneos?

En pos de la resolución de dicho interrogante, procede detenerse en una aseveración en la que los grandes expertos demográficos, tanto a nivel nacional como internacional, coinciden plenamente: la población mundial será doblada en el año 2050. Adviértase que esta inferencia se traduce en que, anualmente, la población adscrita a catorce veces la ciudad de Madrid se añade a la población mundial, o lo que es lo mismo, aproximadamente 45 millones de personas. (Lo que equivale a un valor cercano a la población de España.)

A medida que el planeta adquiere un tinte cada vez más urbano, las metrópolis deben ganar en inteligencia. Con el fin de manejar este proceso de urbanización a gran escala, emerge la necesidad de encontrar nuevas metodologías para gestionar la complejidad del proceso, aumentar el rendimiento de los sistemas urbanos, reducir los costes y, por ende, mejorar la calidad de vida de la ciudadanía.

Con tan prometedor crecimiento en el horizonte, se antoja prioritario otorgar a las grandes urbes el poder de la comunicación. En otras palabras, toda smart city que se precie ha de ser capaz de proveer al ciudadano toda la información que este necesite, desde el estado de la polución hasta el del agua y la electricidad, pasando por el estado preciso del tráfico, los STPU y la disponibilidad de plazas de aparcamiento en los parkings urbanos, por citar algunos ejemplos. De esta forma, es reseñable que no solo la economía se vería beneficiada por la creación de puestos de empleo que mantengan los sistemas de la smart city, sino que a su vez lo propio ocurriría con la salud medioambiental de la urbe, que mejoraría notablemente.

Queda evidenciado, por tanto, el cambio radical al que se verán sometidos los entornos urbanos en el futuro imperiosamente cercano, en el que las tecnologías emergentes adscritas a las TIC gozarán de un protagonismo evidente. El uso de los sensores de bajo consumo, las WSN (Wireless Sensor Networks; Redes de Sensores Inalámbricas), las aplicaciones, tanto en su ámbito web como móvil y las nuevas y prometedoras ramas de la algorítmica computacional, son elementos especialmente llamados a interpretar un rol clave en el importante proceso de transición, contribuyendo a que la existencia de las smart city sea una realidad.

Retomando la pregunta formulada de forma previa al anterior razonamiento, queda especificado que las masivas cantidades de ciudadanos que se asentarán en los núcleos urbanos motivan un rediseño, cuasi total, de los STPU. Entre los problemas asociados a los STPU, se destacan los siguientes, que pueden ser solucionados gracias al Sistema Enjambre, descrito pormenorizadamente en su correspondiente epígrafe en el presente capítulo:

- Carencia de sistemas de respaldo precisos capaces de cuantificar el número de usuarios que esperan en un determinado andén. En los grandes eventos metropolitanos, como puede ser un multitudinario concierto musical o un importante partido de fútbol, es frecuente necesitar de un especial esfuerzo de las autoridades para regular la entrada al Sistema de Trenes Subterráneo, coloquialmente conocido como Metro. De esta forma, se vela por la seguridad del ciudadano, evitando la aglomeración de usuarios en el andén, que podría desembocar en la fatal caída de los mismos al andén. Adviértase que, para evitar tan trágico

desenlace, las autoridades presentes en los tornos de entrada al STPU necesitan estimar el número de personas que están en cierto andén de cara a permitir el paso de más usuarios, o bien estar en contacto por radio con sus homólogos ubicados en la estación. Este proceso es ineficiente, a la vez que dependiente de un ser humano, que, como tal, puede fallar en la estimación. La creación de una metodología precisa que envíe una señal de bloqueo a los tornos hasta que el andén se libere, destaca como un foco de optimización prioritario.

- La integración de todos los factores que pueden interesar al usuario de los STPU, perniciosamente distribuidos actualmente, pueden integrarse en una aplicación móvil. De esta forma, los ciudadanos estarían en disposición de hallar rápida y cómodamente el camino desde la estación A hasta la estación B en base al menor número de transbordos, el menor tiempo de tránsito o la distancia más corta, por citar algunos criterios de búsqueda.
- Inexistencia explícita de sistemas que determinen las rutas más frecuentes o típicas elegidas diariamente por la ciudadanía. El conocimiento por parte de las autoridades gestoras de los STPU de en qué momento una persona sube o baja de un tren supone una puerta a la optimización del sistema, preparando la red de cara a picos de carga en el uso de la infraestructura.
- La publicidad que aparece en las pantallas ubicadas a lo largo de la red de STPU, unida a su homóloga figurante en los monitores empotrados en el interior de los propios vehículos locomotores carece de inteligencia. La extracción de los datos disociados de los usuarios, tales como la edad y el sexo, constituye un importante factor de cara a la dotación de inteligencia de los mismos. Dicha información, unida al estudio de la población que utiliza cierta ruta, supone la herramienta perfecta para mostrar una publicidad acorde a la media de los usuarios. Adviértase que la idoneidad de la publicidad puede, idealmente, participar en la reactivación y el mantenimiento de la economía, puesto que está demostrado que la publicidad efectiva incentiva, tal y como la intuición apunta, el consumo, contribuyendo a la robustez de la economía. Por ejemplo, habiéndose detectado una ruta cercana a un instituto, frecuentada por estudiantes de 13 a 18 años, procedería la inclusión de la publicidad asociada a un nuevo videojuego, mientras que la publicidad adscrita a los nuevos tipos de interés en cierta cuenta de una cadena bancaria puede mostrarse en los monitores de los STPU cercanos a los alrededores del distrito financiero urbano.

A.2.1.1 Ejemplos de Smart Cities

Tal y como el atento lector intuirá tras finalizar la lectura del subepígrafe anterior, el término smart city dista aún de ser aplicado en su totalidad a cualquier ciudad en el presente. A pesar de ello, son dignas de mención ciertas ciudades en las que se han llevado a cabo importantes avances, tanto a nivel nacional como internacional. A saber:

- Málaga: Desde el punto de vista estrictamente informático, destaca el numeroso y variado elenco de aplicaciones (Malaga, 2016) que el ayuntamiento pone a disposición del ciudadano para facilitar sus quehaceres diarios en cuanto a movilidad, ocio, interacción con el ayuntamiento y un largo etcétera. Entre dichas aplicaciones, cabe destacar Málaga Funciona, mediante la cual los ciudadanos podrán notificar incidencias referentes al estado de la ciudad y problemas de tipo general al ayuntamiento y seguir su resolución. Por otro lado, Málaga Ciudad Genial, brinda al usuario un sistema de audio guías creadas por el ayuntamiento que ofrecen información sobre los principales monumentos de la ciudad, horarios, teléfonos de interés, información útil etc. Siguiendo la línea de aplicaciones relacionadas con el ocio, Playas de Málaga ofrece a la ciudadanía un completo informe del estado actual de la zona litoral en dicha ciudad andaluza, así como de información acerca de cómo pueden llegar a ella en función de su posición. Por último en cuanto a aplicaciones, destacan Empleo Málaga y Plan de Empresa, que permiten conocer las ofertas de empleo disponibles en la ciudad, así como ofrecer una guía de emprendedores destinada a ayudar a confeccionar el Plan de Negocio para su comercio, respectivamente.

En lo que al ámbito de la movilidad sostenible, es digno de mención que Málaga participa en varios proyectos en el ámbito nacional e internacional, tales como Zem2All, Green eMotion y Proyecto Victoria, que principalmente aúnan sus esfuerzos en pos de la consecución de un sistema de carga eficiente para vehículos y autobuses eléctricos. A pesar de que esta suerte de sistemas ya existe, estos son eficaces pero no eficientes. Es reseñable que la diferencia entre ambos términos es sutil, pero crucial, pues mientras un sistema eficaz es aquel que consigue todos los objetivos planteados, un sistema eficiente es aquel que los consigue además de la mejor forma posible, de manera óptima.

- Mitchell, Dakota del Sur, EE.UU: Antiguamente una ciudad rural en su práctica totalidad, dejada atrás por aquellos autóctonos que decidían buscar un futuro mejor en otro núcleo urbano, Mitchell ha sido capaz de dar un giro de 360° a su economía, creando una brillante capa tecnológica

sobre su tradicional cultura agraria. La inversión de 40 millones de dólares estadounidenses por parte del Mitchell Technical Institute (MTI; Instituto Técnico de Mitchell), ha resultado una condición *sine qua non* para la concepción de un brillante campus tecnológico, que hace las veces de micro smart city. Dicho campus, dotado de un sinfín de avances tecnológicos [Forb, 2016] que hacen de él una ciudad inteligente a pequeña escala, constituye el lugar de entrenamiento de cientos de técnicos en comunicaciones y datos. Su presencia, unida al voto de confianza inicial otorgado por parte de las empresas presentes, ha convertido a Mitchell en un importante centro de actividad tecnológico de la zona, que a su vez ejerce un poderoso efecto de imantación hacia los talentos en el mundo de las TIC (Tecnologías de la Información y de las Comunicaciones; Information and Communications Technologies).

- Ipswich, Queensland, Australia: A pesar del escepticismo inicial surgido por parte de los organismos públicos, el Plan de Desarrollo Tecnológico (2011) de cara a 20 años para incrementar la población en casi 300.000 personas se tornó en un éxito rotundo. Partiendo de un rediseño cuasi-total de su centro, la inclusión de tecnologías orientadas a la mejora de la seguridad pública (Smart Security) supuso un importante atractivo para los dubitativos compradores de viviendas. Este importante factor, unido estándares de calidad tan respetuosos con el medio ambiente que hacen de esta ciudad el núcleo urbano más sostenible en Australia, han logrado que las previsiones del censo para el año 2031 estén cada día más cerca de ser una realidad.
- Río de Janeiro, Brasil: La asignación de la Copa del Mundo de Fútbol de 2014, unida a la de los Juegos Olímpicos de 2016, ha otorgado una oportunidad única a Río de Janeiro para reinventarse. La cuantiosa inversión recibida no sólo ha permitido a la conocida ciudad brasileña a solucionar gran parte de los ampliamente conocidos problemas en su infraestructura de transporte público, sino que además le ha permitido diseñar sistemas que hacen parecerse, cada vez en mayor medida, a esta ciudad a una auténtica smart city. Estableciendo un símil con Málaga, un ilustrativo ejemplo de los anteriormente citados sistemas se corresponde con Rio Datamine, un sistema de información pública que pone a disposición del ciudadano numerosos y útiles datos de uso cotidiano. Por otro lado, y dado que una smart city sólo puede ser tal si no se despreocupa por la alfabetización de sus ciudadanos, se han construido cuarenta centros educativos gratuitos en el mismo número de barrios marginales. Los factores anteriormente expuestos, unidos al descubrimiento de las fosas de petróleo cercanas a la costa de Río de Janeiro, han provocado que esta ciudad de la costa brasileña recibe

actualmente el doble del capital extranjero que estaba acostumbrada a percibir.

Figura XCVI: Panorámica de Río de Janeiro tras la inversión orientada a la consecución de una *smart city*

Una vez expuestas varias ciudades como ejemplos ilustres en cuanto a la consecución de la denominación *smart city*, y, tal y como el atento lector ya habrá vislumbrado, no existe aún ninguna ciudad en la que el término ciudad inteligente sea una completa realidad; bien las ciudades enfocan sus esfuerzos en la creación de un ecosistema de aplicaciones puramente relacionadas con la información de las mismas, o bien se enfocan a la sostenibilidad medioambiental, entre otras opciones.

En cuanto a la optimización de los STPU se refiere, destaca que la única ciudad de entre las descritas con algún sistema desarrollado en tal sentido es la andaluza Málaga, con su aplicación para la geolocalización de paradas de autobús, así como de su tiempo de llegada. Cabe destacar que dicha aplicación, a pesar de ser increíblemente útil, no constituye un poderoso aliado de Smart Mobility desde el punto de vista del STPU, aunque sí desde la perspectiva ciudadana. Por otro lado, es digno de mención que no se debe incluir a Río de Janeiro en dicho ámbito, puesto que las mejoras llevadas a cabo en su infraestructura de STPU tras las inversiones anteriormente descritas tienen como único objetivo el de dar un servicio aceptable y disminuir a su vez los retrasos, en detrimento de la dotación de inteligencia a sus STPU.

Una vez evidenciada la carencia y desnudez de los proyectos de optimización relacionados con los STPU, se destaca que tal factor pernicioso será exhaustivamente desmenuzado en el siguiente subepígrafe, que a su vez servirá

como justificación de la acuciante necesidad social en la mejora y obtención real de las smart cities.

A.2.1.2 Justificación

El 6 de Mayo de 2015, se redacta en Madrid el informe de la Confederación Española de Organizaciones Empresariales (CEOE) "*Acciones Prioritarias para el Desarrollo de las Smart Cities en España*", más en concreto por la labor del Comité de Smart Cities adscrito a dicho órgano empresarial. Dicho documento es, probablemente, uno de los documentos públicos que mejor señala la necesidad imperiosa de mejora y consecución real en el ámbito de las smart cities, evidenciando la desnudez en el estado del arte referido a la transición de un núcleo urbano a smart city. Por otro lado, se extrae que existe un amplio elenco de ámbitos en los que se deben realizar investigaciones exhaustivas para que el concepto de smart city sea una auténtica realidad.

A continuación se presentan varias conclusiones extraídas de dicho informe, que, dada la acuciante carencia de sistemas capaces de hacer frente a los retos enfrentados por la sociedad contemporánea, justifican plenamente la necesidad de investigar la dotación de inteligencia a los núcleos urbanos:

- En la página número cuatro del citado informe de la CEOE, se destaca que *"Este marco de sostenibilidad y eficiencia que debe envolver a las Smart Cities, tiene una relación directa con otros ámbitos clave, como son los de la Energía, la Movilidad y el Medioambiente y, de forma más particular, con la eficiencia energética, la gestión de residuos (separación y conservación), el ciclo integral del agua (reducción de pérdidas), la rehabilitación energética de edificios existentes, las redes de climatización, la gestión de zonas verdes urbanas y la gestión eficiente de la movilidad de las personas y el transporte de mercancías en términos de sostenibilidad, seguridad y accesibilidad (local, nacional e internacional)."* Por tanto, como conclusión rectilínea del fragmento anterior, se infiere que debe llevarse a cabo un notorio esfuerzo en la gestión eficiente no sólo de los STPU, sino del control de los mismos, alcanzando la eficiencia (y no la eficacia) perseguida, razón que vuelve a demostrar la necesidad de entidades como el *Sistema Enjambre* presentado como línea de ampliación de la presente Tesis Doctoral.
- Seguidamente, en la quinta página del informe, se lee: [Existen] *"Escasez de indicadores para la recogida de medidas adecuadas"*. De nuevo, es reseñable que tal Sistema Enjambre captará millones de medidas que dotan de inteligencia a los STPU disponibles en la ciudad objetivo.

- Por otro lado, la séptima hoja del documento establece los factores clave sobre los que ha de pivotar una Smart City: *"En este sentido, y de forma más particular, entre las oportunidades y beneficios con incidencia sobre la sociedad que han sido identificados como elementos asociados al desarrollo de las Smart Cities pueden encontrarse los siguientes (clasificados en línea con las seis dimensiones de las Smart Cities identificadas en el documento "Mapping Smart Cities in the EU" elaborado por el Parlamento Europeo): 2. Movilidad: [Se necesita un] "Conocimiento en tiempo real sobre incidencias, disponibilidad de aparcamiento, mejora de la eficiencia en la gestión de tráfico y de la gestión servicios públicos de transporte, mejora en la seguridad, etc."*. El presente punto respalda nuevamente el Sistema Enjambre propuesto, evidenciando así la carencia de sistemas de control adscritos a los STPU.
- En una línea análoga, es de vital importancia la información descrita en la vigesimotercera página del informe de la CEOE, más en concreto en el epígrafe *"Movilidad"*, el cual justifica de nuevo el sistema anteriormente mencionado: *"Junto con los sistemas, servicios y edificios inteligentes, es el transporte el factor que logra satisfacer los requisitos de movilidad de los ciudadanos, mejorando su calidad, disminuyendo el consumo energético y aumentando la sostenibilidad."*. Huelga reseñar, por ende, la vital importancia que la CEOE, órgano autoritario en la materia en cuestión, asigna a la existencia de un óptimo estado adscrito a los STPU.
- Reforzando la idea anteriormente expuesta por parte de la entidad, se establece que ciudades inteligentes serán: *"...Aquellas en las que los ciudadanos usen principalmente el transporte colectivo y donde el usuario pueda acceder en tiempo real a la información sobre accidentes, atascos, situación del transporte público, etc."* Además, en la misma línea se destaca que *"...La movilidad en las ciudades es un problema cada vez más acuciante y por ello esta iniciativa es una de las más implementadas en el ámbito de las ciudades inteligentes. El concepto de movilidad se refiere a la sostenibilidad, la seguridad y la eficiencia de las infraestructuras y sistemas de transporte, así como a la accesibilidad local, nacional e internacional."*
- De nuevo, la CEOE evidencia su consciencia con respecto a que *"Uno de los mayores problemas en el ámbito de la movilidad es la congestión del tráfico, que tiene un impacto negativo muy considerable en la calidad de vida de la ciudad tanto por la disminución de la productividad, como por el empeoramiento de la calidad del aire, así como por la contaminación acústica que conlleva."* Las mejoras obtenidas con el presente sistema permitirían mejorar el transporte público urbano, con lo que numerosos usuarios podrían decantarse por él en vez de por el vehículo propio, factor

que contribuiría a aligerar el tráfico. Esta máxima se sintetiza en la siguiente aseveración: *"Por todo ello, la gestión eficiente del transporte de personas y mercancías es uno de los ejes de actuación de la ciudad inteligente, con los objetivos de satisfacer las necesidades crecientes de movilidad de los ciudadanos en el entorno urbano y contribuir a la reducción de emisiones contaminantes y de ruido, mediante un sistema intermodal y conectado a la infraestructura inteligente de la ciudad."*

Una vez expuestas y justificadas las anteriores razones, y a propósito de los sistemas de movilidad urbanos, de especial importancia a fin de lidiar con el crecimiento exponencial de la población, surge un interrogante; ¿cuáles son los principales elementos de movilidad a tener en cuenta en una smart city, así como los datos de interés relacionados? Según el citado informe de la CEOE, órgano autoritario en la materia, se destacan los seguidamente descritos:

- Sistemas Inteligentes de Transporte (SIT): [Dícese de los que realizan la] *"Integración de múltiples soluciones tecnológicas para mejorar la seguridad, la movilidad y la eficiencia de los diferentes medios de transporte en el entorno urbano mediante la gestión del tráfico en tiempo real, de los medios de transporte de viajeros y de los aparcamientos."*
- "Sistemas de información para los usuarios de transporte público", simplificación e integración de los procesos de pago, sistemas de puntos electrónicos de venta, que faciliten la intermodalidad, otorguen información adicional sobre el servicio, etc.
 - Dado que la seguridad y la privacidad son factores vitales en prácticamente cualquier sistema informático personal, y como se recalca en la página vigesimoquinta, epígrafe *"Seguridad y Privacidad"*, se destaca que la comunicación y los datos involucrados en la infraestructura del Sistema Enjambre, utilizado como ejemplo a lo largo del presente epígrafe, serán cifrados, así como a la hora de su almacenamiento. Las técnicas criptográficas pertinentes serán aplicadas de ser requeridas, velando en todo momento por salvaguardar los derechos asegurados por la Ley Orgánica (15/1999 de 13 de diciembre) de Protección de Datos de Carácter Personal en pos de los ciudadanos.
 - Finalmente, siguiendo la taxonomía establecida por la CEOE, se destaca que la presente Tesis Doctoral engloba, por razones obvias, los conceptos de Smart Mobility (Movilidad Inteligente), Smart Safety (Seguridad Inteligente) y Smart Accessibility (Accesibilidad Inteligente), mientras que de forma colateral Smart Employment (Empleo Inteligente, que será requerido para el

mantenimiento del sistema), Smart Tourism (Turismo Inteligente, que será atraído como resultado de la mejora en la Red de Transporte Público), y Smart Education (Educación Inteligente), siendo este último tipo fruto de la elevada capacidad de extrapolación del sistema; las balizas que actúan a modo de dispositivo sensor pueden extrapolarse a un modo de "balizas de reconocimiento" que permitan saber si, por ejemplo, los alumnos están en clase durante las horas estipuladas, dando lugar a un estudio exhaustivo de las tasas de abandono escolar. Los distintos ámbitos que pueden ser estudiados en base a la extrapolación del sistema propuesto están, casi únicamente, acotados por la imaginación.

Continuando con el ejemplo del Sistema Enjambre, y una vez justificada la necesidad acuciante de la investigación y el desarrollo, se presenta a continuación un área muy a tener en cuenta en el ámbito de los núcleos urbanos inteligentes, teniendo especialmente en cuenta el crecimiento exponencial de la población mundial.

A.2.1.3 Control de Masas

El Estadio Olímpico de Madrid, anteriormente conocido como La Peineta coloquialmente, y como Metropolitano de manera oficial, constituye un imponente y ambicioso estadio multiusos ubicado en Madrid. Dicho estadio, actualmente en las fases finales de edificación y futura sede del Club Atlético de Madrid a partir del año 2017, disfrutará de la capacidad de albergar entorno a 72.000 aficionados, además de una cantidad imprecisa de espectadores ubicados en los 94 palcos VIP adicionales. A continuación se presenta una imagen del interior del novedoso estadio:

Figura XCVII: Interior del estadio Metropolitano

En cuanto a su conexión con los STPU se refiere, el Metropolitano estará conectado al sistema de Metro de la Comunidad de Madrid mediante la estación Estadio Olímpico, perteneciente a la línea 7 del Metro de la misma ciudad, en el sector de San Blas-Canillejas. Dicha estación ostenta el privilegio de ser una de las más grandes del sistema de Metro madrileño, gozando de andenes de 160 metros de largo y cuatro vías. Como el sagaz lector ya intuirá, la confluencia de los dos elementos anteriormente descritos desemboca en una galopante problemática, y es que, ¿cómo pueden transportarse 72.000 personas de manera eficiente?

Figura XCVIII: Estación de la Línea 7 del Sistema de Metro de Madrid; Estadio Olímpico

De forma previa al abordaje del proceso de búsqueda de una solución a tal interrogante, ha de recordarse, una vez más, la sutil pero crucial diferencia existente entre un proceso eficaz y uno eficiente: Mientras que el primero consigue los objetivos propuestos (transportar a las personas), el segundo consigue los objetivos propuestos de la mejor manera posible (llevar a cabo el transporte de forma óptima).

Dado que, a tenor de los datos amablemente suministrados por EMT Madrid (<http://www.emtmadrid.es/>), el aforo de la anteriormente citada estación es de aproximadamente 500 personas en cada andén, y teniendo en cuenta que el tiempo medio de paso entre trenes es de 5 minutos, pudiendo albergar los 500 pasajeros que esperan en el andén, puede calcularse el tiempo medio de desalojo del Estadio Olímpico de Madrid en base a la siguiente fórmula:

$$\text{Tiempo}_{\text{desalojo}} = (\text{Aforo}_{\text{estadio}} - \text{Público}_{\text{Transporte privado}}) / \text{Capacidad}_{\text{trenes}} \times \text{Frecuencia}_{\text{trenes}}$$

Supóngase el ejemplo práctico que acontecerá quincenalmente en el presente estadio; un partido de fútbol del Club Atlético de Madrid. En su inicio, los aficionados llenarán el estadio de forma paulatina, acudiendo al estadio de forma variable desde las horas anteriores al comienzo del encuentro. No obstante, el verdadero problema logístico acontece al mismo tiempo que el colegiado del encuentro entone el pitido final, pues los equipos de seguridad del STPU y las

autoridades se verán obligados a coordinar la en absoluto desdeñable cantidad de 72.000 personas en el peor escenario.

Tomando en cuenta un conservador supuesto práctico en el que 22.000 personas utilizan medios de transporte alternativos para llegar al Estadio Olímpico de Madrid, tales como un autobús o un vehículo propio, las autoridades y los equipos de seguridad y logística del STPU aún deben lidiar con media centena de millar de personas que tendrán como objetivo la estación Estadio Olímpico ubicada junto a La Peineta. El tiempo aproximado de desalojo puede hallarse fácilmente sustituyendo los valores anteriormente esgrimidos, otorgados amablemente por EMT Madrid, en la fórmula presentada:

$$\text{Tiempo}_{\text{desalojo}} = (72.000 - 22.000) / 1000 \times 360 = 18.000 \text{ s}$$

Es decir, se infiere que el tiempo total de desalojo teórico sería de 18.000 segundos, o lo que es lo mismo, 300 minutos (5 horas).

Entre los muchos escenarios problemáticos que plantea este supuesto, el atento lector, a buen seguro, ya habrá localizado el quizá más crítico: Si 50.000 personas acuden en tromba a la estación de Metro anteriormente citada, la aglomeración en los andenes, unido a la enorme cardinalidad de los aficionados, desembocará, irremediabilmente, en tragedia, pues alguien puede caer fatalmente a las vías de tren.

Llegados a este punto, destaca la prácticamente única metodología existente en la actualidad en relación a la gestión de dicho escenario, eficaz pero no eficiente. A pesar de existir ligeras variaciones de la misma, todas parten del mismo principio: Los equipos de seguridad y logística de la estación de Metro, con ayuda de las autoridades policiales competentes, frenan a la multitud en la entrada a la estación de Metro, una vez calculan que el número máximo de individuos ha cruzado los tornos de entrada. Cuando reciban la señal de que el tren de Metro ha partido de la estación, de nuevo volverán a permitir el paso a los andenes a una cantidad de personas que, aproximada e idealmente, será igual al aforo máximo de la estación. Una variante de mayor seguridad adscrita al escenario anterior se corresponde con la existencia de un equipo adicional en la entrada al propio andén, situados en una posición accesible una vez superados los tornos de entrada a la estación de Metro, que pueden frenar a los aficionados en caso de que haya existido un error de cálculo por parte del primer equipo logístico y el andén esté llenándose peligrosamente. De esta forma, un equipo en cierto modo redundante puede hacer las veces de salvavidas del protocolo anteriormente citado.

Tal y como la lógica apunta, a pesar de que el protocolo anterior sea eficaz, este dista enormemente de ser eficiente, presentando una carencia evidente: Durante

las cinco horas necesarias para el desalojo del estadio en el supuesto práctico anterior, el personal logístico y las autoridades pueden cometer un fallo humano y permitir la entrada a los tornos de un aforo superior a la capacidad máxima del andén. Este pernicioso escenario podría desembocar en el fatal desenlace que precisamente se pretende evitar, por lo que destaca la necesidad imperiosa de introducir un sistema informático, que, de forma matemática, permita el acceso a la estación a exactamente un número de aficionados idéntico al aforo del andén, alcanzando la ansiada efectividad del sistema de desalojo del estadio. En tal sentido, el Sistema Enjambre presentado como ejemplo de ampliación de la Tesis Doctoral en este mismo capítulo tendría la siguiente estructura:

- Cada estación consta de una mota de smart dust o, en términos coloquiales, baliza, que hará las veces de sink (sumidero, nodo maestro) en el ámbito de una WSN.
- Dicha baliza estará en posesión de comunicarse con los teléfonos móviles o smartphones de los aficionados, que deberán descargarse la aplicación pertinente o, para mayor seguridad, esta debería estar instalada de facto en el dispositivo móvil. Tales elementos interpretarán el rol de balizas de tipo esclavo.
- Una vez la baliza adscrita a la estación detecte que el número de personas posicionadas en el andén iguala al número correspondiente al aforo máximo del andén, esta enviará una señal que bloqueará los tornos de entrada a la estación, impidiendo el paso de más usuarios. Tal cálculo podrá llevarse a cabo mediante un proceso de envío-recepción de mensajes, en el que los smartphones (balizas esclavo) de los aficionados enviarán la señal que define su presencia a la baliza maestro.
- Cuando el tren parta de la estación con los aficionados que esperaban en el andén, la baliza maestro detectará que el número de personas existentes en el andén habrá disminuido, por lo que enviará otra señal que desbloquee los tornos de entrada a la estación, permitiendo de nuevo el acceso al siguiente bloque de aficionados. En este momento, se repetirá el algoritmo descrito hasta que el proceso de desalojo de los aficionados sea efectivo.

Es reseñable que, en el desarrollo del algoritmo, existen ciertos parámetros a ajustar tras la ejecución de un experimento de campo: Dado que los usuarios del STPU necesitan de un determinado tiempo de desplazamiento desde la ubicación de los tornos hasta el andén de la estación, es lícito introducir un cierto retardo en el cálculo llevado a cabo por el algoritmo. De esta forma, el estudio del sistema en la propia estación permite ajustar ciertos parámetros adscritos al proceso algorítmico anteriormente descrito, en pos de lograr un óptimo carácter en el

mismo. Tales ajustes, relacionados con el *factor de seguridad* del algoritmo, permitirían variar dicho factor en función de los datos arrojados por los experimentos de campo, en pos de salvaguardar la integridad de los usuarios. Adviértase que, dado que existen vidas humanas en juego, el presente mecanismo ha de ser considerado como un sistema informático *crítico*.

Por otro lado, en cuanto al control de masas se refiere, destaca la casi obligada emergencia de un interrogante; y es que, ¿qué ocurre si el número de personas que se desplazan en transporte privado es mínimo? Tal escenario desembocará en el crecimiento descontrolado del tiempo de desalojo del estadio adoptado como ejemplo, por lo que es fundamental llevar a cabo un estudio estadístico, fruto de la experiencia y la recolección de datos, de cara a la posible configuración, por parte de las autoridades, de rutas de apoyo alternativas. Así, el sistema de gestión del Metro puede disminuir la frecuencia de paso de los trenes a la finalización del evento deportivo, a la vez que introducir rutas de autobús especiales ad hoc, que trasladen a los usuarios a diversos puntos de confluencia de rutas o intercambiadores. Tomando como ejemplo la ciudad de Madrid, Atocha y Avenida de América son claros ejemplos de objetivos en lo que respecta a la flota de autobuses que partirá desde el estadio.

De forma paralela, es fundamental en cuanto al control de masas la definición de rutas típicas de uso por parte de los usuarios. Persiguiendo la finalidad de delimitar tales rutas, el estudio de los datos a recolectar se antoja obligatorio, por lo que se debe estar en posesión de conocer cuándo una persona sube o baja de un determinado tren. Este conocimiento puede obtenerse gracias al sistema informático anteriormente descrito, pues cuando un usuario llegue a una estación, su smartphone indicará lo propio a la baliza maestro ubicada en el andén, ocurriendo lo opuesto cuando el usuario abandone la misma. De este modo, aparece una nueva necesidad adscrita al ámbito del sistema propuesto: las motas maestro han de ser capaces de enviar sus datos a un CCO (Centro de Control y Operaciones), que recopile toda la información del sistema, constituyendo una puerta al análisis matemático de la misma.

La consecución del objetivo anteriormente descrito permite la definición de rutas típicas tras atender al comportamiento de los usuarios, siendo un poderoso aliado en cuanto a la optimización del sistema se refiere. Así, se posibilita el cálculo y la detección de picos de carga en el STPU, por lo que el sistema de Metro puede incrementar el número de trenes disponible en aquellos periodos denominados como hora punta. Adviértase que este factor no sólo incrementa la efectividad del sistema, sino que a su vez puede motivar a los usuarios a utilizar el STPU en detrimento del transporte privado. Dicho cambio en el comportamiento se traduce en un impacto positivo hacia el medio ambiente, pues las perniciosas emisiones de CO₂ se verían radicalmente minimizadas. La siguiente figura demuestra,

empírica y visualmente, la diferencia que acontece cuando doscientas personas eligen el STPU en detrimento del transporte privado:

Figura XCIX: Comparativa de vehículos utilizados en el transporte de 200 personas

Por otro lado, no debe olvidarse uno de los factores fundamentales en el ámbito social contemporáneo: El marketing. Puesto que la economía constituye el motor por antonomasia de la sociedad, es vital que exista un nutrido flujo monetario en la misma. Entendiendo por marketing el conjunto de estrategias empleadas para la comercialización de un producto y para estimular su demanda, es fundamental orientar dichas estrategias hacia el sector de la población adecuado. Tomando como escenario los STPU, en concreto el sistema de Metro suburbano, esta tarea se antoja ciertamente complicada a tenor de la disparidad de perfiles en cuanto a los usuarios del mismo, que pueden observarse en la siguiente figura:

Figura C: Diagrama de sectores adscrito al perfil de los usuarios de STPU por edad y sexo

Como puede apreciarse, y a pesar de que puede establecerse que existe igualdad en cuanto al uso del sistema de Metro en función del sexo, esta característica no ocurre en cuanto al uso de dicho servicio en función de la edad. De esta forma, surge la necesidad de mostrar, en los monitores que impregnan la red de Metro, anuncios publicitarios en función del perfil del usuario medio en la zona. Así, puede considerarse, por ejemplo, la presentación de publicidad relacionada con videojuegos en un área cercana a cierto instituto o la adscrita a

una tarjeta bancaria de nuevas características en el Metro cercano al distrito financiero. El uso de esta táctica, unido al sentido común (como por ejemplo mostrar la nueva equipación del Club Atlético de Madrid en la estación Estadio Olímpico en detrimento de la estación Santiago Bernabéu), constituye una poderosa sinergia, capaz de impulsar la economía al mejorar sensiblemente la precisión de las estrategias de marketing.

Ahora bien, ¿cómo puede lograrse, desde el punto de vista técnico, dicha mejora en la precisión de la publicidad? Tal misión puede ser implementada a través de la aplicación para smartphones de los usuarios, mencionada anteriormente, existiendo un importante factor a tener en cuenta: La protección de los datos (y por ende, la privacidad) de los usuarios.

En este sentido, y afortunadamente, la legislación es clara y precisa. Más en concreto, la Ley Orgánica 15/1999, de 13 de diciembre, de *Protección de Datos de Carácter Personal*, supone la principal herramienta a la hora de respetar el sistema legislativo vigente en la actualidad.

Dicha ley, afecta a todos aquellos datos de carácter personal registrados en soporte físico que los haga susceptibles de tratamiento, y a toda modalidad de uso posterior de estos datos por los sectores público y privado. Adviértase que, en términos legales, es un *dato de carácter personal*, “*cualquier información concerniente a personas físicas identificadas o identificables*” (Artículos 2 y 3 Ley Orgánica de Protección de Datos, LOPD en lo sucesivo). De esta forma, es reseñable hacer énfasis en la idea que subyace sobre la definición anteriormente citada, y es que se considera como *dato de carácter personal* a cualquier información que permita realizar una correspondencia, bidimensional e inequívoca, entre el dato y el individuo. El Documento Nacional de Identidad, (DNI), es un claro ejemplo de información que establece dicha correspondencia inequívoca.

Por tanto, a fin de evitar esta problemática legal y ética, pues la ley secunda el itinerario seguido por la lógica en este caso, se presenta el concepto de *dato disociado*. El artículo 5.1, apartado e) del *Real Decreto 1720/2007*, por el cual se aprueba el Reglamento de desarrollo de la *Ley Orgánica de Protección de Datos Personales* [Gob, 2018], define los datos disociados como “*aquellos que no permiten la identificación de un afectado o interesado*”. Adviértase que, en esta casuística concreta, no podría llevarse a cabo la correspondencia inequívoca presente en los datos de carácter personal [Prot, 2018]. Tal aseveración puede demostrarse empíricamente mediante el sentido común, pues es, por ejemplo, imposible identificar exactamente quién es un varón de 30 años, al ser dicho dato correspondiente a un gran sector de la población.

Por tanto, teniendo en cuenta los artículos citados, se infiere que no se estará dentro del ámbito de aplicación de la LOPD cuando se esté tratando únicamente datos disociados que no permitan la identificación del afectado, por lo que dicha ley es respetada. Desde el punto de vista económico, destaca que esta diferenciación resulta de gran importancia cuando se plantea la posibilidad de ceder datos, ya que el artículo 11 de la ley en cuestión sólo permite la comunicación de datos a un tercero si se cumplen una serie de requisitos. Por ejemplo, suponiéndose el hipotético caso de una empresa que entregara a otra un fichero de datos que identificaran claramente a sus clientes, con el objetivo de la realización de un estudio de mercado que determinara las zonas de más venta de un producto, estaría realizando una cesión de datos. Como es evidente, esta cesión podría tener problemas si la primera empresa no cuenta con el consentimiento previo de sus clientes. No obstante, el artículo 11.6 de la LOPD establece que *“si la comunicación se efectúa previo procedimiento de disociación, no será aplicable lo establecido en los apartados anteriores”*. De tal manera, si la primera empresa comunica solamente aquellos datos disociados relativos al número de ventas realizadas función de, por ejemplo, la edad de los compradores, no estaría incurriendo en una cesión ilegal de datos personales, respetando la LOPD.

Finalmente, y como el atento lector intuirá, se establece que la aplicación móvil propuesta en la línea de ampliación del Sistema Enjambre almacenará, únicamente, los datos disociados de la misma. Así, el Centro de Control y Operaciones del sistema recibirá la información disociada (edad, sexo, etc.) por parte de las balizas fijas, estando en posición de conocer, de primera mano, la clase de publicidad que debe mostrar en las respectivas zonas adscritas al sistema de Metro suburbano. Adviértase que la presente característica permitiría no sólo llevar a cabo infinidad de estudios estadísticos que permitan mejorar la adecuación del sistema en función de sus usuarios, sino que a su vez constituye un importante activo económico, al contar con gran cantidad de datos disociados (y por ende, legales) adscritos a la población.

A.2.1.4 Análisis de las alternativas de resolución existentes

Tal y como se expuso previamente en secciones anteriores adscritas a la presente Tesis Doctoral, destaca, a nivel europeo, la presencia de Málaga, Dublín, Ámsterdam y Lyon como ejemplos cercanos al concepto de smart city. No obstante, y a pesar de los encarecidos esfuerzos dedicados a la mejora de dichas ciudades por parte de los organismos competentes, es reseñable que existen ciertas carencias en las mismas, tal y como se detallará en el presente epígrafe.

Tras un exhaustivo análisis de las funcionalidades existentes en dichas urbes con respecto (especialmente) a los STPU, destacan las siguientes, presentes en la práctica totalidad de ciudades enumeradas:

- Información de tiempos de llegada online, vía SMS y a través de paneles informativos.
- Sistema de información por voz para personas invidentes.
- Existencia de mapas que permiten conocer la localización de las paradas de autobús/metro más cercanas a la ubicación actual del usuario.
- Sistemas de organización de ruta.
- Red WiFi (*Wireless Fidelity*) gratuita. (Únicamente en Málaga.)
- Servicios de pago de la tarifa del STPU a través de los dispositivos móviles de los usuarios.

Como puede apreciarse, y a pesar del carácter oportuno de las funcionalidades anteriormente especificadas, destaca la carencia de ciertos aspectos especialmente deseables: En primer lugar, tal y como se especificó en el epígrafe anterior, sería especialmente deseable poder medir el número de personas que se ubican en el andén en determinado momento. De esta forma, se posibilitaría el bloqueo de los tornos de entrada a la estación en caso de que el número de personas situadas en el andén se aproxime a la constante de aforo asociada al mismo. En este caso, así como en los escenarios adscritos a la avería de un tren, podría informarse a los usuarios de las diversas rutas alternativas disponibles.

Tal y como el atento lector intuirá, la posibilidad de llevar a cabo una medición constante del número de personas en el andén permitiría incrementar de forma notoria la seguridad del sistema, pues evitaría la caída de usuarios a las vías, eliminando, por ende, un fatal desenlace. Con respecto a los aspectos relacionados con la seguridad del sistema, destaca el existente sistema FLIR [Flir, 2018] (<http://www.flir.es/traffic>). Dicho sistema, especialmente concebido para su uso en tranvías que no transitan de forma subterránea, se compone de un conjunto de cámaras termográficas que, fundamentalmente, evitan accidentes graves y daños en la infraestructura detectando vehículos que se han parado en medio de los pasos a nivel. Por otro lado, el sistema FLIR está en posesión de detectar personas situadas en las vías del tren. A continuación se presenta una figura descriptiva de las posibilidades de detección de las cámaras FLIR, que en cierto modo recuerda al sistema de rescate basado en TensorFlow presentado a su vez como ampliación de la presente Tesis Doctoral:

Figura CI: Ejemplo de detección de situaciones de índole problemática por el sistema FLIR

Por otro lado, en los aspectos referidos a la optimización de los STPU, existe la imperiosa necesidad de poder definir rutas típicas de los usuarios para su uso interno por parte de los organismos competentes. De esta manera, sería especialmente deseable estar en posesión de conocer cuándo un usuario sube y baja del tren adscrito al sistema de Metro, de tal forma que el sistema podría recolectar un jugoso conjunto de datos. Un pertinente estudio estadístico sobre tal información, permitiría calcular picos de carga en el sistema, por lo que los valores atípicos en cuanto a un uso intensivo de los STPU por parte de la población podrían ser abordados mediante la incorporación de nuevos trenes al sistema y, por ende, la reducción de la frecuencia de paso entre trenes. Actualmente, se destaca que no existe ningún sistema estándar de índole similar, por lo que la concepción de una herramienta que aglutine las anteriormente citadas características constituiría un punto de inflexión en el acercamiento de un núcleo urbano al concepto puro de smart city.

Tomando como ejemplo nuevamente la Comunidad de Madrid, es reseñable la existencia de una aplicación móvil que otorga información especialmente útil al usuario de los STPU. Dado que el software en cuestión constituye la alternativa de resolución existente más sobresaliente, se procederá a su análisis en el presente epígrafe. A pesar de que dicha aplicación, denominada Metro de Madrid Oficial [Metro, 2018], ofrece información de valor al usuario, existen ciertos aspectos claramente susceptibles de mejora, tal y como se describirá a continuación. Seguidamente, se ofrece una captura de pantalla asociada al menú de la aplicación en cuestión:

Figura CII: Screenshot del menú principal asociado a la aplicación *Metro de Madrid Oficial*

Como se aprecia en la figura, la aplicación ofrece cuatro opciones en su menú principal: En primer lugar, *Líneas y Estaciones* permite obtener información de primera mano acerca de las diversas líneas que componen la Red de Metro de Madrid (que pueden comprobarse en [Madrid, 2018]), así como el estado de la circulación en las mismas, tal y como se muestra seguidamente:

Figura CIII: Screenshot de la sección *Líneas y Estaciones* de la aplicación *Metro de Madrid Oficial*

A pesar de la evidente utilidad de la información que figura en la captura de pantalla anterior, destaca que, en caso de avería o situación anómala en alguna estación del STPU, la aplicación únicamente indica la existencia del pernicioso escenario, sin referenciar exhaustivamente las alternativas de las que dispone el transeúnte. Tal característica es especialmente deseable, puesto que en el hipotético caso de que una estación no sea transitable por algún motivo, los usuarios deberían estar en posesión de conocer, de forma ágil, cuál es la alternativa ofrecida por las autoridades competentes para suplir la incapacidad de uso de la ruta estándar. A modo de ejemplo, es típico que los organismos de gestión del STPU establezcan rutas alternativas de autobuses, que recorren la ruta aproximada que lleva a cabo el tren adscrito al Sistema de Metro. La información exhaustiva que describa el itinerario de dichas rutas, debe ser un objetivo de una aplicación definitiva.

Por otro lado, destaca una de las más útiles secciones del menú de la aplicación en cuestión, denominada *Trayectos*. Su intuitiva interfaz figura seguidamente:

Figura CIV: Screenshot de la sección *Trayectos* de la aplicación *Metro de Madrid Oficial*

Atendiendo a la figura anterior, el lector puede comprobar cómo la aplicación permite calcular el trayecto deseado desde una calle o una estación hasta otra homóloga de cualquier tipo. Esta flexibilidad dota de una fuerte robustez a la aplicación, pues el usuario está en posesión de calcular la ruta desde, por ejemplo, una estación hasta una calle en concreto y viceversa, no estando sujeto a realizar dicha búsqueda entre dos calles o dos estaciones.

En cuanto al botón *Opciones* que ofrece la aplicación, se destaca que este permite poder realizar la búsqueda seleccionando una hora en concreto para realizar el itinerario, pues dicho factor influye en la hora de llegada. Dicha variación acontece debido a la diferencia en las frecuencias de paso de los trenes durante el día, mayores en la denominada *hora punta*. Adviértase que esta opción es especialmente beneficiosa a su vez desde el punto de vista estadístico, puesto que si las búsquedas realizadas por el usuario se envían a la Central de Control y Operaciones (CCO), se posibilitaría el análisis predictivo de la cantidad de

usuarios esperados en cierto recorrido y en una hora concreta. Como el atento lector a buen seguro intuirá, esta información hace posible, a su vez, el cálculo y la predicción de los picos de carga con los que el sistema ha de lidiar. Continuando la descripción de dichas opciones, es digno de mención que la fecha en la que se recorrerá el itinerario puede ser, a su vez, seleccionada, así como un tipo de ruta que puede variar entre el recorrido más rápido, de menor longitud y de menor número de transbordos. Seguidamente, se presenta una figura referente a las opciones anteriormente descritas, en la que destaca la versatilidad de la aplicación en tal sentido:

Figura CV: Screenshot de la sección Opciones (de Trayecto) de la aplicación Metro de Madrid Oficial

Finalmente con respecto a la aplicación en cuestión, y una vez seleccionadas las pertinentes opciones de Origen, Destino, Fecha del Itinerario y Hora, el usuario está en posesión de llevar a cabo el cálculo de la ruta. Una vez realizado, la aplicación mostrará una pantalla análoga a la similar, en la que se detalla el número de estaciones a transitar y el número de transbordos a acometer, así como el tiempo estimado del viaje:

Figura CVI: *Screenshot* del trayecto a recorrer calculado por la aplicación *Metro de Madrid Oficial*

Como se especifica previamente, y a pesar de las evidentes fortalezas de esta aplicación, destacan, a grandes rasgos, las siguientes carencias, deseables en la misma en pos de ayudar al proceso de transición de un núcleo urbano a una smart city:

- Carencia de un sistema de definición de rutas típicas por parte del usuario: La inclusión de dicha funcionalidad permitiría al usuario agilizar los procesos de selección de ruta, pudiendo albergar rutas guardadas en la propia aplicación. Así, a modo de ejemplo, el usuario podría directamente seleccionar la ruta Trabajo, Casa, Estadio Polideportivo, etc.
- Registro del usuario en la aplicación: La captación de los datos disociados del usuario por parte del software interpreta un papel fundamental en la recopilación de datos estadísticos que contribuyan a mejorar la eficiencia de los STPU. Por otro lado, dicho conocimiento constituye una puerta de

entrada a la presentación de una publicidad acorde a los transeúntes en las distintas zonas que componen el STPU.

- Inexistencia de comunicación con un Centro de Control y Operaciones (CCO): La conversación fluida entre la aplicación y el CCO constituye un factor *sine qua non* para el incremento de la inteligencia adscrita al STPU. La aplicación debe poder comunicarse con las balizas adscritas a una WSN, de cara a calcular si el número de personas en el andén es superior o cercano al límite seguro y bloquear los tornos de entrada al mismo en tal caso. Esta información es susceptible de un análisis estadístico exhaustivo, que permitiría preparar el sistema para lidiar con picos de carga en cuanto al número de usuarios.

Por tanto, las anteriormente citadas mejoras se proponen a su vez como funcionalidades en la aplicación móvil a desarrollar como posible línea de ampliación del Sistema Enjambre, a su vez línea futura de la presente Tesis Doctoral.

Una vez descrita la herramienta anterior, es reseñable que, en términos generales, y tal como apunta el informe de la CEOE exhaustivamente analizado en los primeros epígrafes del presente documento, los sistemas de conversión de un núcleo urbano a smart city y las técnicas de control de masas en dicho sentido brillan por su ausencia. En contraposición, existen numerosos grupos dedicados al acercamiento de las principales ciudades europeas al concepto de smart city. Uno de los más ambiciosos, el Proyecto *EUROCITIES* (<http://www.eurocities.eu/>), constituye una red formada por unas de las más importantes ciudades europeas: Barcelona, Birmingham, Frankfurt, Lyon, Milan and Rotterdam. Los miembros de dicho proyecto, concebido en 1986, típicamente se corresponden con los dirigentes de los gobiernos municipales de cada ciudad. Actualmente, el proyecto anteriormente citado se enfrenta a una ambiciosa expansión, en la que se pretende aglutinar a los ayuntamientos de más de 130 ciudades europeas y 40 ciudades participantes fuera del viejo continente.

El proyecto *EUROCITIES* destaca, principalmente, por constituir un nexo de unión entre los mandatarios de las ciudades involucradas, de cara a convertirse en una plataforma de conocimiento compartido e intercambio de ideas. De esta forma, destaca uno de los principales objetivos del proyecto: La concepción de un marco de smart city europea estándar. De esta forma, los principales esfuerzos están encaminados al desarrollo de un sistema de transición de los núcleos urbanos a smart cities, aplicable, en especial, en el entorno europeo.

En cuanto al ámbito de la smart mobility se refieren, el proyecto *EUROCITIES* promueve la movilidad urbana sostenible, apuntando a la concienciación ciudadana de que los pequeños actos individuales son la llave a la consecución

de un gran bien común a nivel medioambiental. En este sentido, el proyecto en cuestión se centra en la reducción del volumen creciente de tráfico urbano, el cual desemboca en un amplio espectro de problemáticas, tales como la congestión del tráfico, los accidentes y la creciente presencia de la contaminación acústica y la polución, factores que convergen irremediablemente en un negativo impacto no sólo al entorno, sino a la salud de los ciudadanos. De esta forma, el proyecto *EUROCITIES* persigue la mejora, en todos los sentidos posibles, de los STPU, otorgando a los ciudadanos un medio de transporte perfectamente capaz de competir con las alternativas privadas.

A pesar del interesante carácter de los objetivos anteriormente citados, así como de las propuestas concebidas, es digno de mención que no existen aún proyectos de transición a smart city tangibles en el ámbito de *EUROCITIES*. Este pernicioso factor se ve más cubierto en el ámbito del (<https://ec.europa.eu/programmes/horizon2020/>), Programa Horizon 2020 [Horizon, 2018] financiado por la Unión Europea. Dicho programa, consistente en un Programa Marco abreviado como H2020, fija como plazo el año 2020 como un punto de inflexión en los principales núcleos urbanos europeos, llamados a convertirse, en un gran porcentaje, en smart cities desde el concepto puramente etimológico.

Por otro lado, el Programa H2020 tiene como objetivo fundamental la consecución de un triple pilar, para el que la Unión Europea dispone de un presupuesto de 76.800 millones de euros, en absoluto desdeñable:

1. Abordar los retos sociales de primer orden: Dado que el impacto de la crisis económico-financiera ha acrecentado la distancia entre la clase media y la clase alta europea, el Programa H2020 persigue la reducción de tan perniciosa distancia mediante la mejora de las características adscritas a los contratos actuales.
2. Promover el liderazgo industrial en Europa: Reforzar la imagen del viejo continente como adalid de la vanguardia de corte industrial, supone la puerta a la creación de empleo estable y de calidad.
3. Incrementar la excelencia de la base científica europea: Tal y como puede consultarse en *Academic Ranking of World Universities* (Clasificación Académica de las Universidades del Mundo, <http://www.shanghairanking.com>), únicamente dos universidades europeas, correspondientes con Cambridge y Oxford, figuran entre las diez mejores en el año 2015, siendo el resto de universidades que ostentan dicho privilegio de origen estadounidense. La mayor presencia de universidades europeas en dicha clasificación constituye el tercer pilar del Programa H2020.

En cuanto a su filosofía de trabajo, H2020 persigue la integración de numerosas fases vitales en el devenir de un proyecto, desde la creación de conocimiento y el diseño del mismo hasta las actividades más puramente comerciales del mismo: investigación básica, desarrollo de tecnologías involucradas, demostraciones a los inversores, prototipos de fabricación, innovación social, transferencia de tecnología, *proofs of content* (pruebas de concepto), normalización, apoyo a las compras públicas pre-comerciales, capital de riesgo y sistema de garantías que garanticen la *Tasa Interna de Retorno* (TIR) del proyecto en el mínimo tiempo posible.

Destaca a su vez, en el ámbito de H2020, la inclusión del Instituto Europeo de Innovación y Tecnología (EIT), como centro de generación de actividades de investigación, formación y creación de empresas. H2020 no sólo consta de un plan de acción claramente delimitado, sino a su vez goza de una sólida base estratégica que persigue los siguientes objetivos:

- Potenciar el desarrollo de las tecnologías y las aplicaciones necesarias para mejorar la competitividad europea: Tal y como se especificó previamente, H2020 cuenta con una ambiciosa inversión para el desarrollo de tecnología clave para la industria adscrita a las TIC. Se destacan, en este aspecto, las actividades orientadas a la participación de las PYMES (Pequeñas y Medianas Empresas) en proyectos colaborativos de retos sociales, para los cuales se prevé una financiación del 20% del presupuesto existente; 15.360 millones de euros. El anteriormente citado montante pretende financiar el ciclo completo de vida de los proyectos propuestos, desde la evaluación del concepto y su viabilidad al desarrollo, la demostración y posterior replicación en el mercado y el alcance de apoyos para la comercialización con servicios de ayuda para rentabilizar la explotación de los resultados. Adviértase que el enfoque descrito se corresponde con una estrategia bottom-up, de vital importancia en el ámbito de la ingeniería por numerosos aspectos, entre los que destacan los beneficios adscritos a la aplicación del paradigma Divide y Vencerás (*Divide and Conquer*) y la posibilidad de reutilizar módulos ya implementados en el ámbito de un proyecto en distintos homólogos.
- Dotar a la ciencia de origen europeo de un factor de excelencia, reforzando la posición de la Unión Europea en el panorama científico-tecnológico mundial: El presupuesto adscrito al ámbito de H2020 posee, entre otros objetivos, el incremento de la dotación del Consejo Europeo de Investigación (ERC). Dicho organismo tiene por misión la de subvencionar a investigadores europeos de élite, además de velar por la ampliación del campo de las tecnologías futuras cubriendo todos los sectores del ecosistema TIC.

- Identificar, delimitar e investigar los grandes problemas que afectan al ciudadano europeo: En concreto, y tal y como apunta el informe de la CEOE, la atención de H2020 se centra en una decena de áreas fundamentales en una calidad de vida superior: salud, alimentación, agricultura, ciencias del mar, energía, transporte, clima, materias primas, sociedades inclusivas y seguridad. Los proyectos de mejora en dicho ámbito de H2020 pretenden resolver problemáticas concretas del ciudadano europeo, entre los que destacan la protección informática y la concienciación tecnológica, el envejecimiento galopante de la sociedad y el establecimiento de una economía eficiente y respetuosa en cuanto a la emisión de Dióxido de Carbono.

De nuevo, y a pesar del quizá más delimitado y trabajado carácter del *Programa Horizon 2020* con respecto al Proyecto *EUROCITIES*, es evidente que ambos se corresponden con iniciativas a largo plazo. La saturación de los núcleos urbanos actuales, unida al preocupante incremento en los niveles de contaminación [Horizonte, 2018], insta a la comunidad científica a actuar con presteza e inmediatez, ya que, como se ha evidenciado a lo largo de este epígrafe, y a pesar de existir aplicaciones *software* de utilidad, la concepción de proyectos sigue siendo una asignatura pendiente en la transición de los núcleos urbanos al más puro concepto de smart city. Los anteriormente citados factores constituyen un factor extra que refuerza la pertinencia e imperiosa necesidad de orientar los esfuerzos investigadores hacia líneas de investigación como las alternativas de ampliación propuestas para la presente Tesis Doctoral.