UNIVERSIDAD POLITÉCNICA DE MADRID -

ESCUELA TECNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

MASTER IN ARTIFICIAL INTELLIGENCE

MASTER THESIS

CLUSTERING TASK ASSIGNMENT - AN ALGORITHM FOR TIME CRITICAL TASK ASSIGNMENT PROBLEMS

Henrique Meireles Valadares

July, 2017

CLUSTERING TASK ASSIGNMENT - AN ALGORITHM FOR TIME CRITICAL TASK ASSIGNMENT PROBLEMS

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TECNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

ARTIFICIAL INTELLIGENCE DEPARTMENT

MASTER THESIS

CLUSTERING TASK ASSIGNMENT - AN ALGORITHM FOR TIME CRITICAL TASK ASSIGNMENT PROBLEMS

Author: Henrique Meireles Valadares

Director: Nik Swoboda

July, 2017

Henrique Meireles Valadares

Madrid – Spain

E-mail: h.mvaladares@alumnos.upm.es
E-mail: henriquemv2003@yahoo.com.br

Overview

This project will provide an overview of assignment problems. In order to understand these concepts, some classical task assignment problems will be introduced. Later on, the definition of time critical task problem will be provided. The most popular approaches to solve task assignment problems will be introduced. Some of these approaches include: Integer programming, Bidding approaches and Field based approaches. Each one of these approaches have advantages and disadvantages.

In the following chapter, we will see how task assignment algorithms can be applied to solve logistics and supply chain problems. In addition, we will see the Multi-Agent Programming Contest that is organized by Clausthal University of Technology(Germany) every year. This contest was designed to motivate research in the field of multi-agent systems. The 2016 contest scenario requires the implementation of time critical task assignment algorithm that can be applied to logistics and supply chain problems.

The Custering Task Assignment (CTA) is a time critical task assignment algorithm that tries to address the flaws of the most popular approaches. The idea is to cluster the groups into tasks with the best candidates(free agents) according to a heuristic. In other words, the place where agents needs to reach(task facility) is considered to be as a cluster centroid and the agents inside each cluster are the candidates to execute each task. The name Clustering Task Assignment has to do with a logical concept in which agents form groups. The algorithm does not take usage of any clustering mathematical approach, for instance the clustering regions are never calculated, they serve only as graphical output to facilitate algorithm understanding.

The following chapter provides an example of the CTA applied to the 2016 Multi-Agent Programming Contest. It describes the whole task assignment process steps by step instantiating agents and tasks with values. In the results chapter, the CTA is measured against CNEP in terms of assignment speed and solution efficiency. It explains why the CNEP was chosen as the comparison target. Besides, this chapter explains the metric of comparisons, scenarios of comparisons and attributes used to generate statistics.

Abstract

Time critical task assignment problems are frequently found in operating systems and flight control applications. Tasks have to be assigned in less than one second, otherwise operating systems don't work properly or catastrophes could happen. For instance, in a flight control real time application, one wrong task assignment decision could result in an airplane accident. To make things worse, these crucial decisions have to be taken in milliseconds, therefore, the efficiency of these kind of algorithm has to be optimized.

In order to address time critical task assignment problems, the Clustering Task Assignment Approach CTA is proposed. The idea is to cluster the groups into tasks with the best candidates(free agents) according to a heuristic. In other words, the place where agents needs to reach(task facility) is designed as a cluster centroid and the agents inside each cluster are the candidates to execute each task. Agents compete with each other by means of an objective function in order to define the assigned task. This heuristic is capable of dramatically reduce the number of calls to the score function, thus assigning groups of agents efficiently to groups of tasks.

The algorithm is compared against the well known Contract Net Protocol(CNET) and outperform it in terms of speed reducing the number of calls to the score function in 400%, the solution quality is 28% better as well. This algorithm was designed to work with time critical applications in which agents have to move from one place to another and when the score function is very costly. However results show that it can also be applied to time non critical applications because the solution quality found are outstanding in comparison to other approaches.

Contents

Overview **ii**

Abstract **iii**

Contents **iv**

List of Tables **vii**

List of Figures **viii**

List of acronyms **ix**

1 Introduction **1**

1.1 Definition of task assignment problem 1

1.2 Complexity of task assignment problems 2

1.3 Definition of time critical problem 3

1.4 Time non critical problem 4

1.5 Time critical problems 5

1.6 Document structure 6

2 Related Work **8**

2.1 Operations Research Approach 8

2.1.1 Integer Linear Programming (ILP) 9

2.1.2 Multi-Objective Linear Programming 9

2.1.3 Continuous Linear Programming 9

2.2 Bidding Approaches 10

2.2.1 Contract Net Protocol 10

2.3 Field-Based Approaches (FBA) 16

3 Case study **19**

3.1 Logistics 19

3.2 Multi-Agent Programming Contest 19

4 Clustering Task Assignment (CTA) specification 22

4.1 Approach overview 22

4.1.1 Definitions of heuristic terms 22

4.2 Cost function complexity - study case 24

4.3 Algorithm steps 25

4.4 Comparison Contract Net Approach - Extended Implementation 29

4.5 Algorithm complexity 29

4.6 Dynamic Task assignment 30

4.7 Suitable scenarios to apply the approach 31

4.8 Sample of scenario in which the approach could be useful 31

4.9 Scenario in which CTA will perform poorly 31

5 Applications of CTA to the 2016 Multi-Agent Programming Contest 33

5.1 Scenario description 33

5.2 Defining tasks priorities 34

5.3 Define every task as a cluster centroid 34

5.4 Define formula to apply the number of free agents in each cluster 34

5.5 Defining cluster aggregation rule 34

5.6 Defining the candidate to assign task 35

6 Results 36

6.1 How CNET was implemented 36

6.2 Metrics of comparison 36

6.3 Time Metric of comparison 36

6.4 Performance Metric of comparison 37

6.5 Scenarios of comparison 37

6.5.1 Single run scenario: 37

6.5.2 One hundred runs scenario: 38

6.6 Statistics 38

7 Conclusions 41

7.1 Comparisons CTTA/CNEP 42

7.1.1 Comparison in terms of speed 42

7.1.2 Comparison in terms of solution quality 42

7.2 Should people always use CTT instead of CNET? 42

7.3 Achieved goals 43
7.4 Skills acquired 43

8 Future research **44**

References **47**

List of Tables

4.1 Vehicle speeds - Multi-Agent Contest 2016 23

5.1 Vehicle properties - (Coordinates are not perfectly scaled) 33
5.2 Task properties - (Coordinates are not perfectly scaled) 33
5.3 Ratio calculation between Cluster 1 and free agents 35

6.1 Single run - Contract Net 38
6.2 Single run - CTA 39
6.3 Assignment variation between CNET and CTA 39
6.4 Performance and speed comparison CNET/CTA - one hundred runs 40
6.5 Solution quality comparison CNET/CTA - one hundred runs 40

List of Figures

2.1 Forming local auctions: Consider Agent A1 through A5, all requesting to be assigned to Task k. Solid lines between the agents indicate communication links. In the proposed scenario, two local auctions are formed involving agents A1 and A2, and A1, A3 and A5, respectively. Note that A1 participates in both auctions. If bid(A1) > maxbid(A2), bid(A3), bid(A5), then A2, A3 and A5 will switch to a different task, while A1 and A4 will keep requesting Task k. If bid(A2) > bid(A1) > maxbid(A3), bid(A5), then A1, A3 and A5 will switch to a different task. Similarly, other bidding scenarios can be considered and the local auctions are guaranteed to break any ties over Task k.[20] 16

2.2 Field Based Obstacle Avoidance 17

2.3 Field Based Task Assignment - Agent A is more attracted to task 1 rather than task 2 because it is closer to it. If task 1 were busy, it would emit a repulsive field 18

3.1 MAS Contest Monitor 21

4.1 Clustering Task Assignment Overview 24

4.2 Path Destination 25

4.3 Clustering Task Assignment 26

4.4 CTA same number of tasks and agents 27

4.5 Clustering Task Assignment - time step 10 30

4.6 Scenario in which CTA might not perform well 32

8.1 CTTA new feature - step 1 45

8.2 CTTA new feature - step 2 45

8.3 CTTA new feature - step 3 46

List of acronyms

AGV Automatic Guided Vehicle

CTA Clustering Task Assignment

CNET Contract Net Protocol

DSS Distributed Sensing System

FBA Field-Based Approach

FBTA Field-Based Task Assignment

ILP Integer Linear Programming

SMP Stable Marriage Problem

Chapter 1

Introduction

1.1 Definition of task assignment problem

The task assignment problem consists in finding a way to assign certain available resources (machines or people) to perform certain tasks at the lowest cost, assuming that resources can be allocated by multiple tasks, and that each task can be executed by a multiple resources. It is one of the fundamental problems of combinatorial optimization in the branch of optimization or operational research. Task assignment algorithms can be applied to the assignment of tasks to employees, factories to products, sellers to territories, bidders to contracts, etc. With a simple implementation, task assignment can be applied to optimization problems. A task-assignment algorithm seeks an assignment that optimizes a certain cost function, for example, maximum throughput or minimum turnaround time. However, most reported polynomial algorithms yield sub optimal solutions[29]. In general, optimal solutions can be found through an exhaustive search, however in real life, computers do not have unlimited time at their disposal and we do not have an indefinite amount of time to find the optimal solution. Depending on the form of the cost function, assignment problems can be classified as linear or quadratic.

Optimal solutions to the linear assignment problems can be computed in polynomial time using the Hungarian algorithm [15]. The quadratic assignment problem, however, is NP-hard [28] and sub optimal solutions are achieved by means of various relaxations. Approaches are either purely discrete [1], or continuous [31], based on the solution of differential equations that always converge to a discrete assignment.

Assignment approaches can be either online [16], [32] or off-line [10] depending on whether the assignment of agents to tasks is designed to account for changes in the task environment or is solved independently in advance.

Task assignment also aims to enhance the system's performance, typically by reducing the overall execution time. In transport multi-agent task assignment problems, performance is associated to the minimization of costs that agents have to pay to travel around the environment. ([10],[26])

1.2 Complexity of task assignment problems

In other to understand better the complexity of assignment problems, let us have a look at some classical task assignment problems in the literature:

- **Generalized assignment problem** There are a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost and profit that may vary depending on the agent-task assignment. Moreover, each agent has a budget and the sum of the costs of tasks assigned to it cannot exceed this budget. An assignment in which all agents do not exceed their budget and total profit of the assignment is maximized. [4]
- **Linear bottleneck assignment problem** There are a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. All tasks are performed by assigning exactly one agent to each task in such a way that the maximum cost among the individual assignments is minimized.
- **Weapon target assignment problem** Suppose a ground command center or an airborne mission command center has to reassign a set of platforms or reallocate their weapons to a set of targets. Each platform is assumed to carry one or more weapon types, and each target is fully or partially satisfied by one type of weapon served by a platform (referred to as the "source" hereafter). A source is able to serve multiple targets, and we assume it delivers only one type of weapon when it visits a target. A source leaves the starting location, serves different targets and returns to the ending location. The source cannot travel over the distance of its travel capacity (based on available and bingo fuel). An assignment (or task) can be described as assigning a source to targets with proper weapons satisfying the travel capacity limit.[22]
- **Mines and factories problem** Suppose that we have a collection of n mines mining iron ore, and a collection of n factories which consume the iron ore that the mines produce. Suppose for the sake of argument that these mines and factories form two disjoint subsets M and F of the Euclidean plane R2. Suppose also that we have a cost function c : R2 × R2 → [0,), so that c(x, y) is the cost of transporting one shipment of iron from x to y. For simplicity, we ignore the time taken to do the transporting. We also assume that each mine can supply only one factory (no splitting of shipments) and that each factory requires precisely one shipment to be in operation (factories cannot work at half- or double-capacity). Having made the above assumptions, a transport plan is a bijection T : M → F. In other words, each mine m M supplies precisely one factory T(m) F and each factory is supplied by precisely one mine. We wish to find the optimal transport plan, the plan T whose total cost is the least of all possible transport plans from M to F. This motivating special case of the transportation problem is an instance of the assignment problem. More specifically, it is equivalent to finding

a minimum weight matching in a bipartite graph. [3]

- **Quadratic assignment problem** There are a set of n facilities and a set of n locations. For each pair of locations, a distance is specified and for each pair of facilities a weight or flow is specified (e.g., the amount of supplies transported between the two facilities). The problem is to assign all facilities to different locations with the goal of minimizing the sum of the distances multiplied by the corresponding flows. [6]

Most of these problem can not be solved just with a pen and piece of paper. Because the number of combinations of possible values(combinations of agents and tasks) can vary too much. Also, finding the best combinations is not simple, to make things worse, a small change in one variable could decrease dramatically the value in the objective function. Therefore, the use of computer aid is required so that the best combination of variables can be found. In addition, switching values manually is not feasible. Furthermore, most of this problems have applications in the real life and bad task assignment decisions could result in a large financial punishment.

For example, the weapon target assignment problem could be applied to a army that is fighting a war and needs to assign weapons to their squads. The cost of weapon transportation is really high. In addition, assigning wrong weapons to wrong squad can result in useless soldier deaths, and in extreme cases, the loss of the entire war. Therefore, making the best decision as possible is crucial.

The Quadratic assignment problem could be applied to logistics problems in which a company need to transport products between their subsidiaries. The company wants to decrease the consumption of gasoline as much as possible and transport the largest amount of goods as possible. Not defining the best combination of products for loading vehicles and not sending vehicles to the best location as possible could result in high financial loss.

Therefore, the complexity of task assignment problems are usually very high. To make things worse, this complexity can increase according to the required time to provide a solution.

1.3 Definition of time critical problem

Some applications of task assignment problems are real-time because they require response in a small time slice. There are plenty of definitions of time critical problem which are mostly borrowed from real-time applications like: e.g: In real-time applications, the computer is often required to service programs in response to external signals, and to guarantee that each such program is completely processed within a specified interval following the occurrence of the initiating signal. [24]. Real-time programs must guarantee response within specified time constraints, often referred to as 'deadlines'. [27]. The correctness of these types of systems depends on their temporal aspects as well as their functional aspects.

Real-time responses are often understood to be in the order of milliseconds, and sometimes microseconds. A system not specified as operating in real time cannot usually guarantee a response within any time-frame, although actual or expected response times may be given.

As we have seen, most of the definitions are vague in regard to the exact amount of response time to classify a problem as time critical. Therefore, for purpose of classification, we are going to define a **time critical problem** as a problem that requires response in less than one second, any other kind of problem that can return the response in more than one second will be considered **time non critical problem**.

The complexity of time non critical problems is lower than real time applications because the algorithms have more time to provide better solutions through a more consistent and elaborated search mechanism. Although some algorithms do not require a response in critical time, their complexity can also be very high.

1.4 Time non critical problem

To illustrate with more details a time non critical task assignment problem present in the literature: The stable marriage problem (SMP) [18] is a problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences for each element. A matching is a mapping from the elements of one set to the elements of the other set. A matching is not stable if: There is an element A of the first matched set which prefers some given element B of the second matched set over the element to which A is already matched, and B also prefers A over the element to which B is already matched. In other words, a matching is stable when there does not exist any match (A, B) by which both A and B would be individually better off than they are with the element to which they are currently matched.

The stable marriage problem has been stated as follows: 'Given n men and n women, where each person has ranked all members of the opposite sex in order of preference, marry the men and women together such that there are no two people of opposite sex who would both rather have each other than their current partners. When there are no such pairs of people, the set of marriages is deemed stable.'[18]

The SMP is a NP-hard optimization problem although it doesn't require response in real time. In this classical task assignment problem, processing could take longer, which means that the search can last further, resulting in a more reliable outcome. If we apply the SMP to a real life problem in which people are looking up for partner to share an apartment and each one has access to each other profile in a web page. Therefore, people could rank partners by preference but the matches are calculated by the system. The algorithm would need to search in the database the best matches. In addition, this search would take really long if the number of users is large. For this problem, founding a solution in less than one second is possible, however results might not be very reliable. In other words, one couple could be

matched, however with possibilities of finding better matches. Therefore, for this problem, taking from one to ten seconds to find a solution would be acceptable, thus matching this problem as time non critical problem. Another reason to match this problem as a time non critical lies on the decision importance. Since the decision of living of someone is crucial, it is better to spend more time searching for a more reliable match rather than running the risk of matching two people that are more likely to have relationship problems.

1.5 Time critical problems

Differently from the SMP, some task assignment problems are time critical. For instance, how operating system assign tasks to processors. In other words, given a set of K task to be executed on a distributed system of n processors, to which processor should each task be assigned? Besides being an NP-hard problem, it is also important to keep in mind that the user is not willing to wait seconds to execute a program after clicking on a button. Therefore, this is a time critical task assignment problem because near optimal solutions need to be found in less than one second so that operating systems can work smoothly. Near optimal solutions means that sometimes the set of tasks won't be assigned to the best possible set of processors. Most of the time is more important to find near optimal solutions rather than taking much longer to find slightly better solutions. In addition, users wouldn't feel pleased to click on a button and wait seconds until something happens, therefore, it is important that the operating system returns an answer in less than one second. Another very import time critical task assignment problem is the air traffic control problem in which we are interested in scheduling runways and gates for aircraft at a busy airport such as Chicago's O'Hare. At any given time there are planes approaching the airport, landing on runways, loading and unloading passengers and cargo at gates, waiting for departure, waiting for repair, and taxing down runways for take-off. The air traffic controller must take into account all this activity in order to schedule the runways and gates. The controller transmits speeds, headings and altitudes to the planes to satisfy the scheduling. As situations change these instructions may be modified. The objective is primarily to maintain the safety of passengers and equipment and secondarily satisfy performance measures such as minimizing fuel cost and delays and maximizing runway throughput. [8]

The controller's task is most difficult during peak periods when the number of planes and possible varieties of situations is very large. During non-peak hours tasks can be solved with considerably less time and effort. Furthermore, the pattern of peak and non-peak hours is fairly regular throughout the week. For example, at 4am there are few planes arriving at O'Hare. The controller can construct optimal schedules for these planes and can modify them on demand based on weather changes and other uncontrollable events. At 6pm the situation is drastically different. The controller cannot hope to construct optimal schedules on demand. Some form of advanced planning is necessary. Advanced planning requires a

model of upcoming traffic and uncontrollable events in order to get some idea of the possible situations that may require responses at 6 pm. While 4am and 6pm vary greatly in both problem complexity and problem parameters, 6 pm on Tuesday is of the same difficulty level as 6 pm on Wednesday even if the actual combination of planes and other parameters varies. This regularity can be exploited in allocating on-line deliberation time between the 4am and 6pm task.[8]

As we have seen, the complexity of critical problems can be very high. For instance the operating system task assignment problem and traffic control problem are very complicated and require responses in small amount of time. Some bad task assignment decision can lead to financial loss. For example, in the traffic control, if a plane land in a gate that is too far away from the airport ground vehicles, the vehicles would need to move uselessly from one place to another. As a result, wasting fuel. In critical cases, assigning one plane to land in a busy gate could result in a crash which would lead to a terrible catastrophe.

As a consequence of the complexity of time critical problem, the study of this kind of algorithms are very important because this problems are usually very complex and require answers in few time slice. For example, time critical problems could be as complex as the SMP problem, however requiring solutions in less time like real time applications.

In order to address the the problem of returning responses in few amount of time in time critical problems, the implementation of heuristic methods is essential. An heuristic is defined by [21] as : ' Any approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals where finding an optimal solution is impossible or impractical, heuristic methods can be used to speed up the process of finding a satisfactory solution.' An Heuristic is applied when finding an optimal solution is impossible or impractical in a small time slice, heuristic methods are used to speed up the process of finding satisfactory solutions. In other words, heuristics are improvements in the convergence of algorithms(most of them search algorithms), so that in each iteration the solutions found provide large steps in the score function value. This process lead to a faster convergence to a set of optimal solutions.

1.6 Document structure

Chapter 2: Related Work

This chapter will represent most popular approaches to solve task assignment problems like: Contract Net Protocol, Field Based approach and Linear Programming.

Chapter 3: Case study

This chapter will explain some real life problems that require task assignment algorithms. For instance, logistics and supply chain. In addition, one research use case that tries to address this kind of issue.

Chapter 4: Clustering Task Assignment (CTA) specification

This chapter will explain the ideas behind the Clustering Task Assignment (CTA) to solve time critical task assignment issues: The full algorithm description, criterion for heuristics, rules and formulas applied.

Chapter 5: Applications of CTA to the 2016 Multi-Agent Programming Contest

This chapter will provide an example of one task assignment run with the CTA algorithm in the 2016 Multi-Agent Programming Contest. The steps will be explained step by step, replacing variables with agents and facilities properties values.

Chapter 6: Results

This chapter will contrast the proposed algorithm techniques with the very well-known Contract Net Protocol. Some statistics and graphics will be provided as metric of comparison.

Chapter 7: Conclusions

This chapter analyzes the range of scenarios provided in chapter results and concludes if the proposed algorithm outperform the contract net protocol. Besides, will describe the advantages and disadvantages of the new algorithm.

Chapter 8: Future research

This chapter will explain the strategies addressed to reduce CTA algorithm flaws.

Chapter 2

Related Work

2.1 Operations Research Approach

Operations research is a discipline that deals with the application of advanced analytic methods to help make better decisions. It is often considered to be a sub-field of applied mathematics. Operations research is often concerned with determining the maximum (of profit, performance, or yield) or minimum (of loss, risk, or cost) of some real-world objective. In other words, operations research are designed to solve optimization problems. It tries to fulfill these goals employing techniques from other mathematical sciences, such as mathematical modeling, statistical analysis, and mathematical optimization. Similarly to heuristics, operations research arrives at optimal or near-optimal solutions to complex decision-making problems.

Operations research can address optimization problems present in myriads of different fields of studies, like:

- Globalization: globalizing operations processes in order to take advantage of cheaper materials, labor, land or other productivity inputs
- Routing, such as determining the routes of buses so that as few buses are needed as possible.
- **Supply chain management**: managing the flow of raw materials and products based on uncertain demand for the finished products.
- **Task Assignment Problems**: such as the problems described in chapter 1, section: Complexity of task assignment problems.

In this approach problems have to be modeled in three steps:
1)Define required variables to model the problem
2)Define the objective function or score function(which is the goal the problem is trying to solve). This function always tries to maximize or minimize something.
3)Define restrictions(the possible range of values the variables can assume).
Lastly, the algorithm has to iterate in order to change possible variable values so that the

objective function is maximized or minimized. The way in which provided solutions iterate relies on the technique applied.

Operations research stress their methods of solving optimization problem though Mathematical Programming. These methods are applied according to the type of objective function: Lineal or not lineal. Inside Lineal problems, there are:

2.1.1 Integer Linear Programming (ILP)

This problem is the one in which all variables are integer and there is just a single objective to maximized or minimized. This methods can be implemented to solve most classical task assignment problem stated in chapter 1:

1) Quadratic assignment problem

2) Generalized assignment problem

3) Linear bottleneck assignment problem

This method is applied to solve plenty of task assignment problem in real life too. For examples, in these articles ([5] , [23]). In [5], the author describes an optimization problem in which a company wants to make its supply chain department more efficient, so that its profits increase. The problem consists in : Each actor can perform a maximum number M of tasks, each task can be assigned to a subset of actors and a cost is associated to the couple task-actor. Assuming that the tasks are indivisible, the requirements of the assignment problem are the following: i) assigning all the tasks to the actors; ii) assigning to each actor no more than M tasks; iii) minimizing the maximum total load of each actor. The assignment problem is solved when all the agents agree on the same assignment.

2.1.2 Multi-Objective Linear Programming

This problem is the one in which all variables are integer. However, there are at least two objectives that can be converging or conflicting. For instance, let us suppose that a manufacturer of chemicals wants to produce on a large scale two types of substances, called A and B. The manufacturer wants to maximize the benefit (objective 1) but at the same time wants to minimize the emitted pollution gases (objective 2) that occurs in the manufacturing process. Therefore, the objectives are conflicting.[9]

2.1.3 Continuous Linear Programming

This problem is the one in which at least one variable is continuous and there is just a single objective to maximized or minimized. One classical task assignment problem that can be solved using this technique is the : **Mines and factories problem**. It differs from other problems in the aspect that the amount ore to be transported should be a discrete value.

Pros and Cons Linear Programming approaches

The disadvantages of Linear Programming approaches are related to any kind of search

approach: getting stuck in local minimal. This method has the advantage of allowing the implementation of heuristics to speed up the search convergence or exploration. Plenty of heuristics have been developed throughout the years to improve linear programming task assignment convergence, for example:([17] , [11]). This is one the most common approach to solving task assignment problems because it has been studied by Operations Research for a long time. In addition, Linear Programming is also applied to solve problems in plenty of different fields of studies.

2.2 Bidding Approaches

Plenty of approaches have been developed across the literature. One of the most popular approaches is the well know Contract Net Protocol (CNET). Differently from ILP, the CNET solve task assignment problems by means of a protocol, it was created by [25] in the 80's.

2.2.1 Contract Net Protocol

The contract net protocol has been developed to specify problem-solving communication and control for agents in a distributed problem solver. Task distribution is affected by a negotiation process, a discussion carried on between agents with tasks to be executed and agents that may be able to execute those tasks. The utility of negotiation as an interaction mechanism can be used to achieve different goals: such as distributing control and data to avoid bottle necks and enabling a finer degree of control in making resource allocation and focus decisions than is possible with traditional mechanisms. [25].

The contract net protocol was designed to solve distributed problems like: traffic-light control, distributed sensing, heuristic search. In Smith's study case, he applies the CNET to a distributed sensing system (DSS). A DSS is a network of sensor and processor agents spread throughout a relatively large geographic area. It attempts to construct and maintain a dynamic map of vehicle traffic in the area. However, the CNET was widely used in the electronic market for buying and selling of goods. It is was also often applied to Multi-agent systems, more specifically to task assignment problems.

When the author mentions a negotiation process, he refers to a bidding approach in which a **contractor** bids for a **manager** job. The **manager or initiator** is responsible for monitoring the execution of a task and processing the results of its execution. A **contractor or participant** is responsible for the actual execution of the task. A **bidding approach** is the one which contractors bid for tasks and the manager determine the most advantageous contract to sign. The criterion of selection are determined by the manager. The negotiation between managers and contractor have the following characteristics:

- Any agent can be a contractor or manager
- It is a local process that does not involve centralized control. In other words, each one of the agents are autonomous in regard to their ability of processing information and

communication.

- There is a two-way exchange of information. In short, manager can communicate with contractor and contractor can send messages to manager.
- Each party to the negotiation evaluates the information from its own perspective. That is, contractor can bid and manager can raise the bid. The decision of accepting this raise is up to the contractor.
- Final agreement is achieved by mutual selection. In other words, manager will analyze each one of the valid bids to make his decision.

The Contract Net Protocol is broken down into 6 steps. These rules have to be applied so that the negotiation process is carried out successfully:

1)Task announcement

An agent that generates a task normally initiates contract negotiation by advertising existence of that task to the other agents with a task announcement message. It then acts as the manager of the task. A task announcement can be addressed to all agents in the net (general broadcast), to a subset of agents (limited broadcast), or to a single agent (point-to-point). The latter two modes of addressing, which we call focused addressing, reduce message processing overhead by allowing non addressed agents to ignore task announcements after examining only the **addressee** slot[25] . This is useful in case the manager only wants to warn free agents about a new job. A task announcement has four main slots:

- **Eligibility specification** is a list of criteria that a agent must meet to be eligible to submit a bid. This slot reduces message traffic by pruning agents whose bids would be clearly unacceptable. In a sense, it is an extension to the addressee slot. Focused addressing can be used to restrict the possible respondents only when the manager knows the ids of appropriate agents. The eligibility specification slot is used to further restrict the possible respondents when the manager is not certain of the ids of appropriate agents, but can write a description of such agents.
- **Task abstraction** is a brief description of the task to be executed. It enables a agent to rank the task relative to other announced tasks. An abstraction is used rather than a complete description in order to reduce the length of the message.
- **Bid specification** is a description of the expected form of a bid. It enables the manager to specify the kind of information that it considers important about a agent that wants to execute the task. This provides a common basis for comparison of bids and enables a agent to include in a bid only the information about its capabilities that is relevant to the task, rather than a complete description. This both simplifies the task of the manager in evaluating bids and further reduces message traffic.

- **Expiration time** is a deadline for receiving bids. We assume global synchronization among the agents. However, time is not critical in the negotiation process. For example, bids received after the expiration time of a task announcement are not catastrophic: at worst, they may result in a sub optimum selection of contractors.

The Common Interagent Language

It is useful to encode slot information in a single high-level language understandable to all agents. The CNET calls this a common interagent language. Such a language, along with a high-level programming language (for transfer of procedures between agents), forms a common basis for communicating slot information among the agents. While the contract net protocol offers a framework that specifies the type of information that is to fill a message slot, it remains the difficult task of the user to specify the actual content of the slot for any particular problem domain. CNET does, however, offer the user some additional assistance. It provides a very simple language, based on an object, attribute, value representation. The language includes a simple grammar, predefined for each slot, and a number of predefined domain-independent terms (e.g., TASK, TYPE, PROCEDURE, and NAME). The representation, the grammars, and the do main independent terms are offered to the user to help him organize and specify the slot information. He must augment the language with domain-specific terms (e.g., SENSOR) as needed for the application at hand. A message that does not have to be understood by many agents (e.g., messages exchanged by a manager and contractor during execution of a contract) can be usefully encoded in a private language. This can reduce both the length of the messages and the overhead required to process them. In CNET, such "private" information is preceded by an "escape" character; this allows private information to be inserted in any message, even one that includes some public information encoded in the normal manner. [25]

2)Task Announcement Processing

In CNET, all tasks are typed. For each type of task, a agent maintains a rank-ordered list of announcements that have been received and have not yet expired. Each agent checks the eligibility specifications of all task announcements that it receives. This involves ensuring that the conditions expressed in the specification are met by the agent. If it is eligible to bid on a task, then the agent ranks that task relative to others under consideration. Ranking a task announcement is, in general, a task-specific operation. Many of the operations involved in processing other messages are similarly task-specific. CNET defines a task template for each type of task. This template enables a user to specify the procedures required to process that type of task.[25]

3)Bidding

At this point, the contract processor is enabled to submit bids on announced tasks. It checks its list of task announcements and selects a task on which to submit a bid. If there is only one type of task, the procedure is straightforward. If, on the other hand, there are a number of task types available, the agent must select one of them. The newest version of CNET selects the most recently received task (older tasks are more likely to have been already awarded). An idle agent can submit a bid on the most attractive task when either of the following events occur: 1) the agent receives a new task announcement or 2) the expiration time is reached for any task announcement that the agent has received. At each opportunity, the agent makes a (task-specific) decision whether to submit a bid or wait for further task announcements. (In the signal task, a potential contractor waits for further announcements in an attempt to find the closest manager.) The agent abstraction slot of a bid is filled with a brief specification of the capabilities of the agent that are relevant to the announced task. It is written in the form indicated by the bid specification of the corresponding task announcement. The agent abstraction slot can also include a number of REQUIRE statements (e.g., REQUIRE PROCEDURE NAME FFT). Statements of this form are used by a bidder to indicate that it needs additional information if it is awarded the task. REQUIRE statements can be made if two conditions are met: 1) the required objects were not preceded by MUST-HAVE terms in the eligibility specification of the task announcement and 2) the objects are transferable; that is, they can be transferred by message. (A procedure falls into this class, but a hardware device does not.) The task template is helpful here. If a agent receives an announcement for a type of task with which it is not familiar (i.e., does not have the template), then it can request the template as a convenient shorthand for the entire set of procedures associated with that type of task.[25]

4)Bid Processing

Contracts are queued locally by the manager that generated them until they can be awarded. The manager also maintains a rank-ordered list of bids that have been received for the task. When a bid is received, the manager ranks the bid relative to others under consideration. If, as a result, any of the bids are determined to be satisfactory, then the contract is awarded immediately to the associated bidder. (The definition of satisfactory is task-specific.) Otherwise, the manager waits for further bids. If the expiration time is reached and the contract has not yet been awarded, several actions are possible. The appropriate action is task-specific, but the possibilities include: awarding the contract to the most acceptable bidder(s); transmitting another task announcement (if no bids have been received); or waiting for a time interval before transmitting another task announcement (if no acceptable bids have been received). This is in contrast to the traditional view of task allocation where the most appropriate agent available at the time would be selected. Successful bidders are informed that they are now contractors for a task through an **announced award** message. The **task specification** slot

contains a specification of the data needed to begin execution of the task, together with any additional information requested by the bidder. [25]

5)Contract Processing, Reporting Results, and Termination

Once a contract has been awarded to an agent. The **information message** is used for general communication between manager and contractor during the processing of a contract. The **report** is used by a contractor to inform the manager (and other report recipients, if any) that a task has been partially executed (an **interim report**) or completed (a **final report**). The **result description** slot contains the results of the execution. Final reports are the normal method of result communication. Interim reports, however, are useful when generator-style control is desired. A contractor can be set to work on a task and instructed to issue interim reports whenever the next result is ready. It then suspends the task until it is instructed by the manager to continue (with an information message) and produce another result. The manager can also terminate contracts with a **termination message**. The contractor receiving such a message terminates execution of the contract indicated in the message and all of its outstanding subcontracts.[25]

6)Negotiation Tradeoffs

Because bids are binding and a agent is allowed to have more than one bid outstanding at a time, a agent may receive multiple awards. These are queued for processing in order of receipt. The cost is potentially slower overall system performance (the load may be less evenly balanced) than would be the case if multiple awards were prevented. If agents could refuse awards, multiple awards could be prevented. However, the cost is at least one additional acknowledgment message per transaction. In some cases, it may be many additional messages (if an award is refused by several bidders). Similarly, if a agent could only have a single bid outstanding, multiple awards could be prevented. However, the cost would be significant delay. Agents would be forced to remain idle until a task announcement had expired to find that their bids had been rejected, and have to start the process again. This could lower overall system performance. The above delay could be reduced in some instances by explicitly informing unsuccessful bidders that their bids had been refused. The cost, however, would likely be a very large increase in message traffic, assuming that there are several bidders per task. In addition, it would only lower the delay for contracts that were awarded before the expiration times of their task announcements were reached. The CNP allows an agent to bid at intervals related to the receipt of task announcements rather than at fixed intervals. It is intuitively appealing, offers a reasonable compromise between message traffic and delay in allocating tasks, and has exhibited good performance in experience to date with CNET. The CNP allows an agent to submit, at most, a single bid at each opportunity. This reduces message traffic and the possibility that a single agent could bid on far more tasks than it could process. For the same reasons, the CNP allows only idle agents to submit bids. Because of

these choices, contracts are not centrally notarized. This further reduces message traffic and maintains the distributed nature of the negotiation process.[25]

CNET complexity

The complexity of the CNET is lineal because the standard CNET documentation only allows that each agent bid for a single task at a time. In other words, if N = number of tasks to assign. The problem complexity equals to (N). Therefore, for a scenario in which 16 tasks have to be assigned, each one of the agents will bid for a single task, the number of calls to the score function will be 16. However, in the standard CNET implementation, task assignment quality can be horrible without an heuristic to determine which task to bid. Therefore, the standard algorithm implementation might be very inefficient in terms of solutions quality. On the other hand, it can be very efficient in terms of assignment time.

CNET improvements

Some improvements were made in this approach across the years. The DynCNET [30] allows an agent to cancel an task in case of fail. This other extended version of CNET [12] permits an agent to bid with multiple clients at the same time until it finds the most suitable client(the one who is more profitable for him). The problem of this approach is that if the score function is costly , assigning task will be costly as well, because every agent bids for each task, resulting in excessive score function calls. The advantage of this approach is the high solution quality, since it guarantees that the most suitable agent gets selected.

One approach that implements bid approach and tries to address this issue is [20]. In this approach each auction is performed among neighboring groups of agents and requires only local communication. This approach increases the CNET complexity from lineal to polynomial because it allow agents to bid multiple times. That is: $n(n + 1)/2$, where n is the number of tasks to assign. For instance, if 16 tasks have to be assigned, in the worse case scenario 136 score function will be executed. As a consequence, the computational cost is cut down dramatically. However, this approach was designed to work with agents that have same features, if agents have different characteristics, modifications to the algorithm would be required.

The tendency of most approaches that tries to modify the CNET is to remove the single bid restriction. This strategy slows down the algorithm speed, however increases solution quality. If this restriction is taken out, CNET could find optimal solutions for some specific scenarios, however it would require an heuristic to determine near optimal solutions if the number of assignments at once is too large. Since, the exclusion of this restriction generates a new optimization problem(finding the best combination of assignments in order to maximize benefits). Without this restriction, CNET can guarantee the optimal set of assignments for the following scenarios:

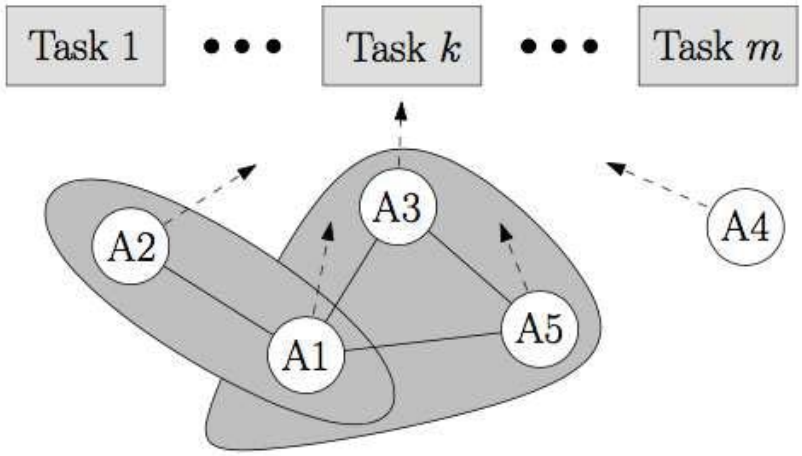

Figure 2.1: Forming local auctions: Consider Agent A1 through A5, all requesting to be assigned to Task k. Solid lines between the agents indicate communication links. In the proposed scenario, two local auctions are formed involving agents A1 and A2, and A1, A3 and A5, respectively. Note that A1 participates in both auctions. If bid(A1) > maxbid(A2), bid(A3), bid(A5), then A2, A3 and A5 will switch to a different task, while A1 and A4 will keep requesting Task k. If bid(A2) > bid(A1) > maxbid(A3), bid(A5), then A1, A3 and A5 will switch to a different task. Similarly, other bidding scenarios can be considered and the local auctions are guaranteed to break any ties over Task k.[20]

*all tasks have priorities

*the number of tasks to be assigned in a single round is not too large.

For examples, if there are 3 tasks with different priorities to be assigned and 3 agents, the number of calls to the score function to determine the optimal set of solutions will be: 3*2*1 = 6. In other words, the 3 agents will bid for the first task, the agent who returns the highest score function value will be assigned the task, for the following tasks, only the two remaining agents will compete for the other tasks. However, if tasks have no priority, we would need to find the best configuration of each agent to each task, so that the sum of score functions values are maximized. In other words, this is an optimization problem itself, the algorithms would need to call the score function: 3*3*3 = 27 times. Therefore, the number of calls to the score function grows exponentially with the number of tasks.

2.3 Field-Based Approaches (FBA)

Typical protocol-based approaches for task assignment such as Contract Net have proven their value, however, they may not be flexible enough to cope with continuous dynamics in the environment. Potential fields is a technique originating from the area of robotics where it is used in controlling the navigation of robots in dynamic environments.

In the beginning of the 80's, potential field methods (PFM) were mainly designed for obstacle avoidance and have gained increased popularity among researchers in the field of robots and mobile robots. The idea of imaginary forces acting on a robot has been suggested by [2] and [13]. In these approaches obstacles exert repulsive forces onto the robot, while the target applies an attractive force to the robot. The sum of all forces, the resultant force R, determines the subsequent direction and speed of travel. One of the reasons for the popularity of this method is its simplicity and elegance. Simple PFMs can be implemented quickly and initially provide acceptable results without requiring many refinements. [14] has applied the potential field method to off-line path planning and [14] suggest a generalized potential field method that combines global and local path planning.

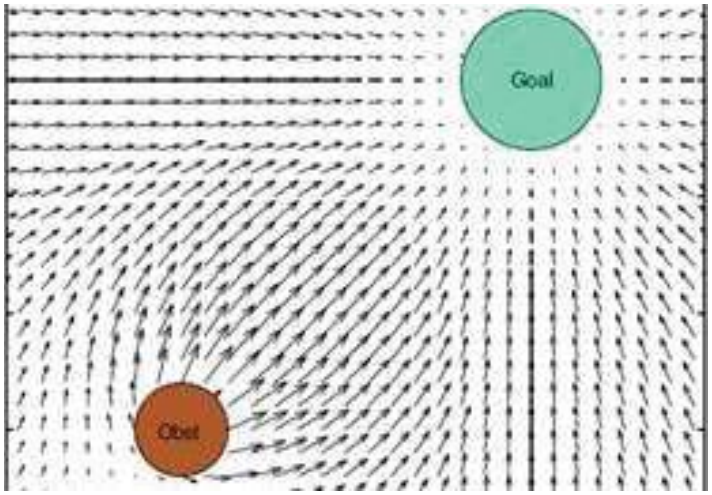

Figure 2.2: Field Based Obstacle Avoidance

Latter on, Field-Based Approaches were exported to task assignment problems. For example, the approach implemented in an Automatic Guided Vehicle (AGV) transportation system [30]. In the field-based task assignment for AGVs, transports emit fields that attract idle AGVs, while competing AGVs emit repulsive fields. Each idle AGV follows the gradient of the combined fields it receives, which guide the AGV towards the pick location of a transport. In other words, the idea works the other way around as obstacle avoidance. That is, for Task Assignment Field-Based algorithms, the obstacle are replaced by task location(the goal where the robot wants to reach), and tasks attract agents instead of repulsing.

As described above, the Field-Based Task Assignment(FBTA) approach works differently from the CNET and ILP because it doesn't use a protocol nor perform search. This aspect can really speed up task assignment process, however this method is completely dependent to distance from vehicle to tasks. If vehicles have different features, this approach wouldn't be able to take into account.

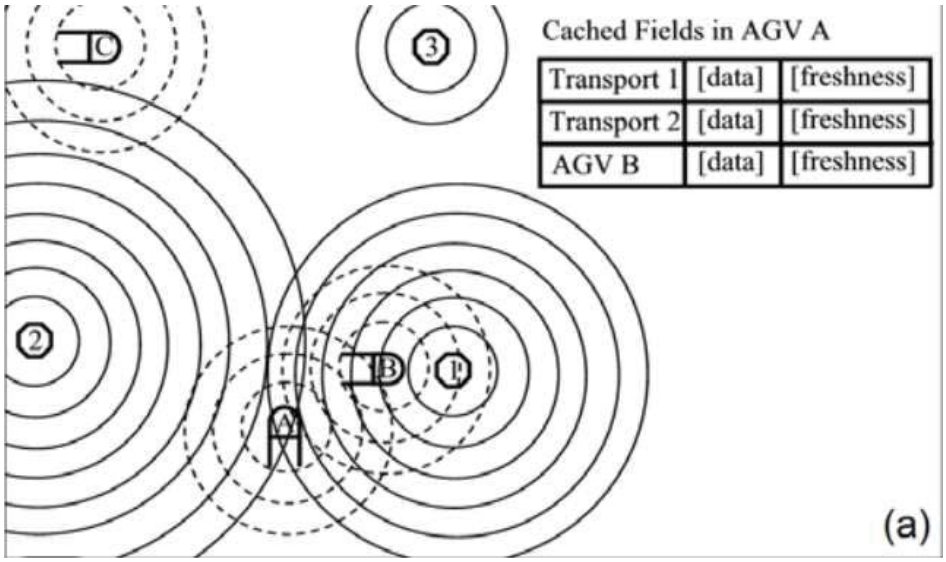

Figure 2.3: Field Based Task Assignment - Agent A is more attracted to task 1 rather than task 2 because it is closer to it. If task 1 were busy, it would emit a repulsive field

Chapter 3

Case study

3.1 Logistics

Logistics is generally the detailed organization and implementation of a complex operation. In a general business sense, logistics is the management of the flow of things between the point of origin and the point of consumption in order to meet requirements of customers or corporations. Large retailer companies such as WalMart and Carrefour have the need for very efficient supply chain services in order to attend client demand. From the consumer perspective we have no idea how complex this problem is. For instance, when we buy a beer at a supermarket , we don't realize that the supermarket staff had to buy the beer from a supplier, the supplier had to purchase the product from an factory, the factory had to buy the basic ingredients before producing and selling it. However, how is this product transported from peer to peer? How does the supplier define the best way to load their trucks and distribute to retailers?

In order to address task assignment issue that could be applied to logistics, researchers from Clausthal University of Technology organize an annual Multi-Agents Contest. This research also attempt to stimulate research in the area of multi-agent system development and programming by:

- identifying key problems.
- collecting suitable benchmarks.
- gathering test cases which require and enforce coordinated action.

3.2 Multi-Agent Programming Contest

The scenario used in 2016 consists of two teams of agents moving through the streets of a realistic city. The goal for each team is to earn as much money as possible. Money is rewarded for completing certain jobs. Jobs comprise the acquisition, assembling, and transportation of goods. These jobs can be created by either the system (environment) or one of the agent teams.There are two kind of jobs: priced and auctioned. A team can accept an auctioned job by bidding on it. The amount of money bid is the reward. If both teams bid, naturally the lowest bid wins. If a job is not completed in time, the corresponding team is

fined. Priced job have a reward defined upfront, that is given to the first team to complete that job. The teams have to decide which jobs to complete and how to do them, i.e. where to get the resources and how to navigate the map considering targets like shops, warehouses, charging stations, storage facilities.

A team consists of different types of agents. The agents differ in their speed, how they move around the city, battery charge, the volume of goods they can carry, and which tools they can employ to craft other goods. Currently we have 4 types: cars, trucks, motorcycles, and drones.

Goods can be bought, crafted, given to a teammate, stored, delivered as part of a job completion, recovered from a storage facility, and dumped. These action may happen at their respective specific locations/facilities. The crafting of an item requires the use of other goods, some which serve as basic materials and some which serve as tools. Since each kind of agent can only handle a subset of the tools, the crafting of some goods require the explicit collaboration of 2 or more agents.

Agents posses a battery charge that gets decreased as they move around from one place to another. They need to make sure they never run out of charge, and therefore should plan their visits to the charging stations accordingly. Moving from one place to another, as well as recharging the battery at a charging station, are actions that are carried out only partially on each step, and may require several steps for completion.

Tournament points are distributed according to the amount of money a team owns at the end of the simulation. To get the most points, a team has to beat the other team, as well as surpass a certain threshold. If a team has a large debt, points are deducted. [19]

Competitors have access to a scenario's monitor which allow them to keep track of simulation. As described in the picture bellow, each team can analyze what is going on at run-time.

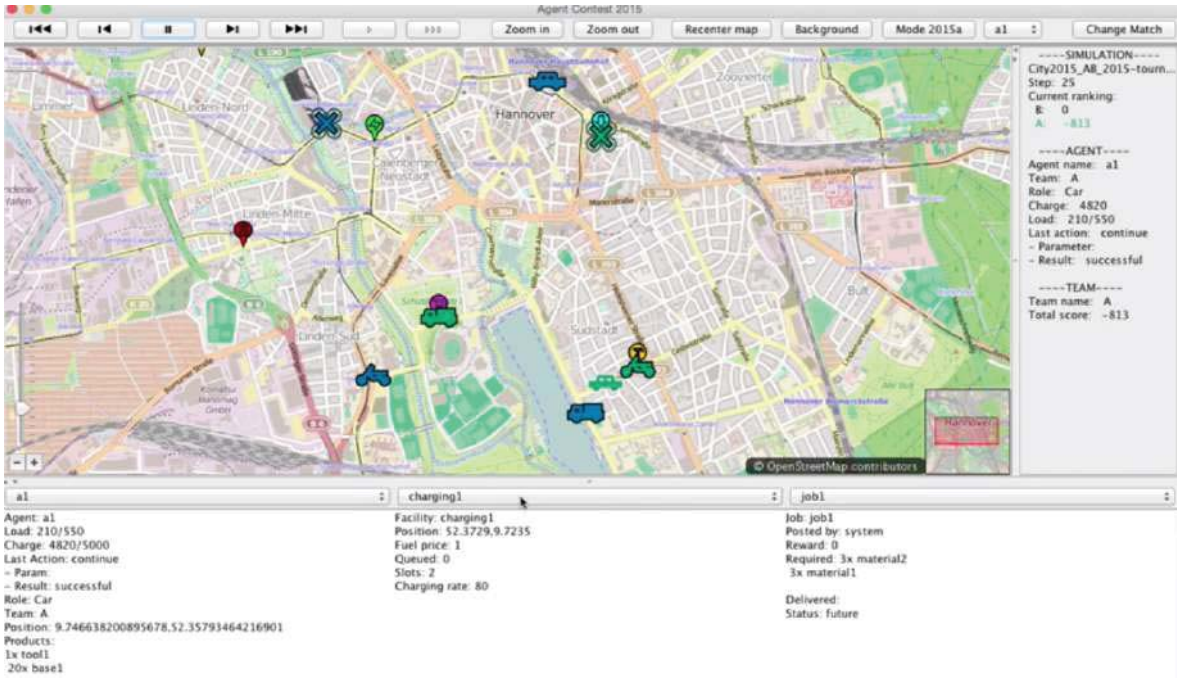

Figure 3.1: MAS Contest Monitor

Chapter 4

Clustering Task Assignment (CTA) specification

4.1 Approach overview

The proposed approach is similar to the FBA because it considers the proximity of one agent to the point of interest to assign tasks. However, sometimes just the proximity to the task is not the best factor to determine whether an agent should fulfill a job or not. Some agents features could be relevant to assign a task as well. In the contest study case, one agent could be closer to a task facility, however another agent who is a little bit further from the facility may arrive at the destination quicker because it is a faster or due to its flying capacity. In order to address this observation, the Clustering Task Assignment Approach is proposed, the idea is to cluster the tasks into groups with the best candidates(free agents) according to a heuristic. In other words, the place where agents needs to reach(task facility) is designed as a cluster centroid and the agents inside each cluster are the candidates to execute each task. The name Clustering Task Assignment has to do with a logical concept in which agents form groups. Figure 4.1, shows 6 free agents, 1 busy agent and 4 destinations in which agents have to reach. The goal there, is to determine the best agents to send to each location. Figure 4.1, shows how agents formations are similar to clusters. However, the algorithm does not take usage of any clustering mathematical approach, for instance the clustering regions are never calculated, they serve only as graphical output to facilitate algorithm understanding. In addition, the name clustering is used because the group formations are similar to clusters and these squads are used to determine which agents are more suitable to each task, therefore reducing the number of call to the score function. Before beginning the heuristic overview explanation, it is important to have a look at some important term definitions.

4.1.1 Definitions of heuristic terms

- **Cluster centroid:** This heuristic is designed to address problems in which agents have to move from place to place. In other words, the cluster centroid is always the place where agents want to reach, they are not exactly the gravity center of the regions highlighted in the following pictures. In figure 4.1, Charging Station, Storage, Shop and Workshop are the 'tasks', which means places where free agents want to reach. Therefore, these four locations are called 'cluster centroids'.
- **Free agent:** is always a not busy agent. An agent who is idle or waiting for a new task

to be assigned. At the implementation level, the agent idleness is determined by an empty action vector. In other words, the agent is not executing any action by that time.

- **Candidates:** are always free agents willing to fulfill a task. Figure 4.1, shows 6 free agents: Drone1,Car1,Motorcycle,Truck,Truck1,Car2. Every free agent is always a candidate to a single cluster and only agents within their cluster will compete for that task. For instance, Drone1 is candidate to centroid Charging Station, Car1 is candidate to centroid Storage, Motorcycle and Truck are candidates to centroid Shop, Car2 and Truck1 are candidates to centroid Workshop. The heuristic is designed this way in order to decrease the number of calls to the score function. As we will see, calculating the route between Truck1 and Charging Station would be a tremendous waste of time, since they are clearly too far away and trucks are very slow. It is obvious that Truck1 is not the best vehicle for reaching Charging Station. It is important to point out that the green colored cross(a drone) next to Storage **is not** a candidate because it is not a free agent. Therefore, it is not inside that cluster.
- **Clustering:** The clusters are defined in the moment when tasks have to be assigned (The formations in figure 4.1 will change every time a new set of task has to be assigned). Clusters are formed by clustering centroid, and position and velocity of candidates. Figure 4.1 shows for clusterings, the clustering whose centroid is Charging station has the candidate Drone1 inside. The clustering whose centroid is Storage has the candidate Car1 inside. The clustering whose centroid is Shop have the candidates Truck and Motorcycle inside. The clustering whose centroid is Workshop have the candidates Truck1 and Car2 inside.
- **Distance:** is given by the euclidean distance between vehicle and task location(cluster centroid).
- **Speed:** For the study case, the vehicle speed is given by number of steps traveled in each iteration.(see table 4.1)

Table 4.1: Vehicle speeds - Multi-Agent Contest 2016

Vehicle	Speed
Truck	1
Car	3
Motorcycle	4
Drone	5

In a realistic application, this value could be the average speed traveled by the vehicle in one hour. This value could also be adjusted according to the roads on which agents are driving. For instance, the average speed inside city with or without traffic, the average speed on a highway.

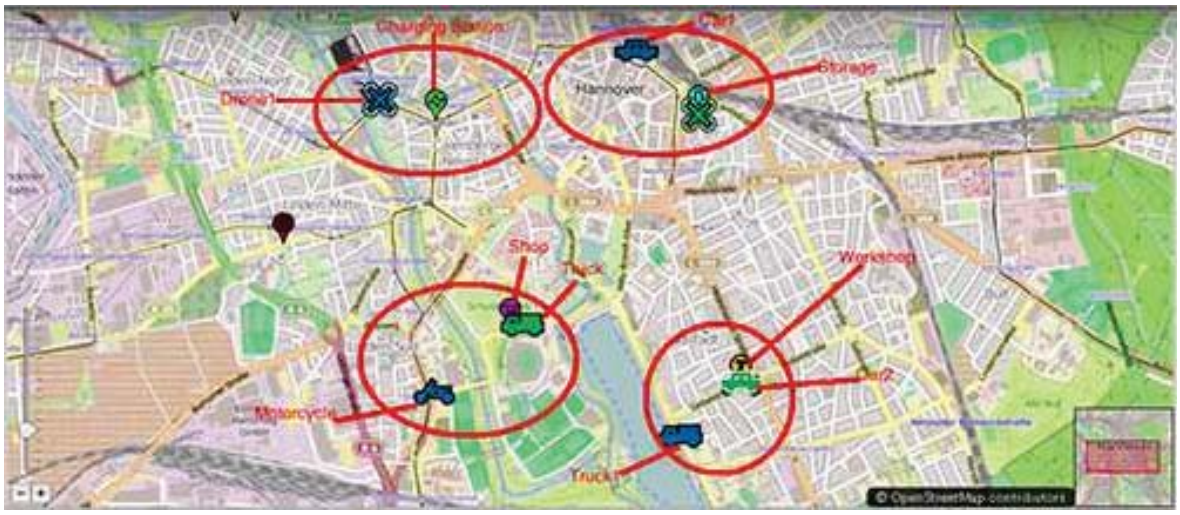

Figure 4.1: Clustering Task Assignment Overview

4.2 Cost function complexity - study case

In order to define the best agent to execute task, a score function is required. For the contest, we can assume that the score function is the cost and time from moving from one place to another, therefore, the score function is defined mainly by two variables:
1) the cost that can be related to the amount of gasoline consumed and also to the cost to sell an item in a shop.
2)The amount of time to arrive to the destination.
For example, let us suppose that one agent has to buy a item at a shop, Shop 1 is closer to the vehicle however sells the required item for a higher price than shop 2 which is further away. It is important to consider that the agent will consume more gasoline to reach shop 2 rather than shop 1. Therefore, identifying which action is better is not simple. In other words, an evaluation in a objective function is relevant to determine which path is best. For the contest study case, the score function is very costly because in order to calculate the gas used, the route between every intermediate point has to be calculated as well. For example, one vehicle wants to go to a shop, however this distance is too far to reach with a single tank. Thus, It is possible that the vehicle has to fill up the tank multiple times before arriving at the destination. This implies in the calculation of the route between the vehicle and intermediate locations until the destination. However these three route are not the only ones calculated, because the score function itself is a optimization problem. In other words, the vehicle has to pass through the least number of charging stations and drive the shortest distance possible to arrive at the destination efficiently.

Figure 4.1, shows the optimal path from the vehicle to a destination. It is important to notice that the optimal path depends on the amount of gasoline the vehicle has. In the following scenario the motorcycle was almost without gas, therefore it had to head to a charging station immediately.

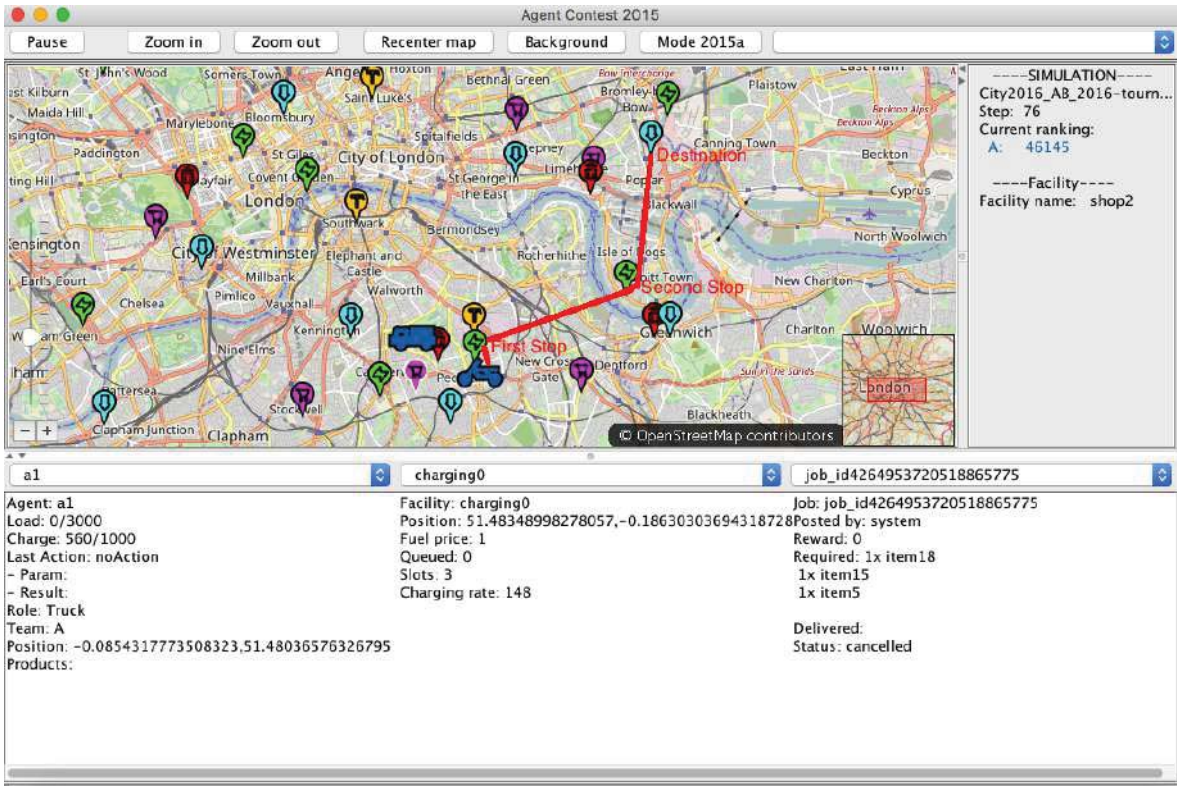

Figure 4.2: Path Destination

To summarize, the score function also has to calculate the fitness of many paths to charging stations, so that the best path from the vehicle to the destination is found. Another important thing to consider, is the fact that if the vehicle runs out of gas, a tow truck has to be called which is a very costly penalty that implies a large cost and time loss.

Having in mind that some score functions are very costly, the proposed algorithm's main goal is to reduce the number of score function calls based on a heuristic. This heuristic is defined by the way agents are clustered as candidates to assign tasks based on their speed and proximity to a goal.

4.3 Algorithm steps

Step 1: **Defining the free agents and the tasks to assign**

The main goal of the heuristic is to define the most suitable agents to perform the provided tasks, taking into account that the tasks are locations where agents want to reach. Whenever a task or group of tasks have to be assigned, free agents are identified in order to define the most suitable agent to perform required tasks. If there is not at least one free agent, all the tasks enter in a queue. The following steps are only executed if there is at least one free agent to execute jobs.

Step 2: **Defining tasks priorities**

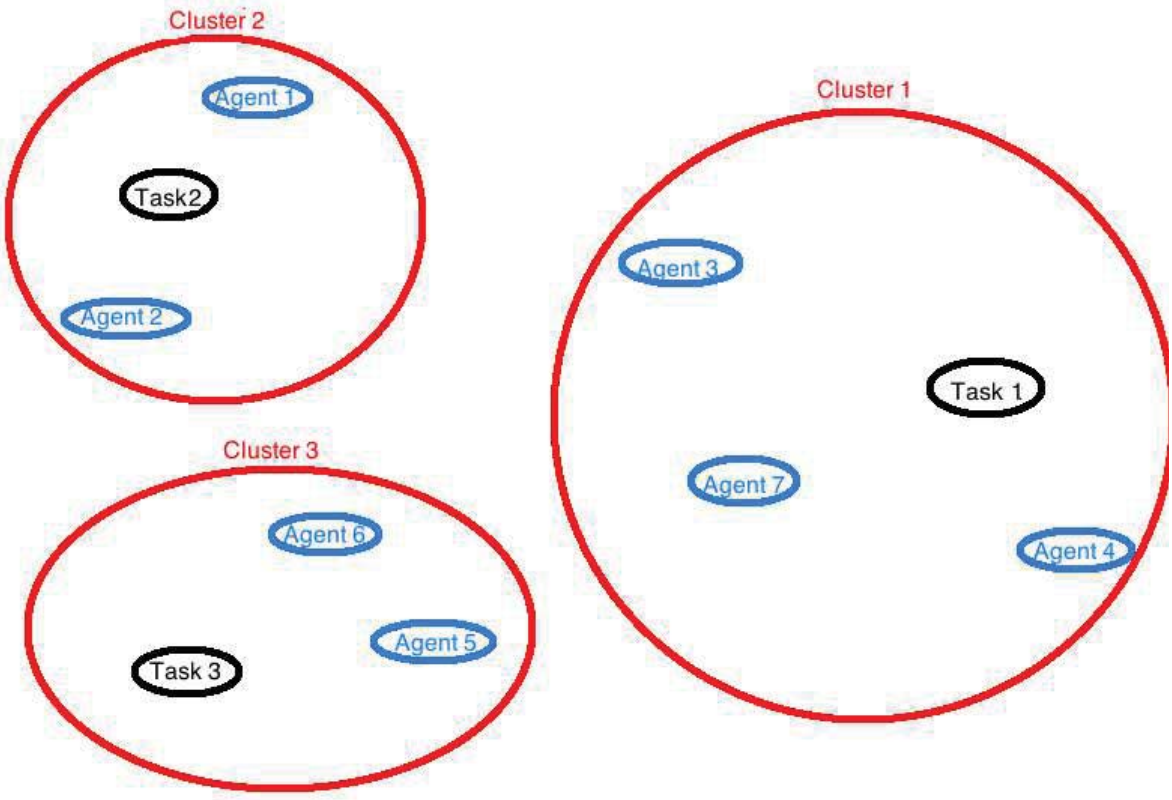

Figure 4.3: Clustering Task Assignment

Tasks have to be sorted by priority. The higher the priority, the higher the probability of having more agents competing for that specific task. If tasks have no priority, the tasks are ordered by creation time. If some tasks have priority and others not, the prioritized tasks are ordered and the remaining tasks enter in the end of the queue sorted by creation time.

Step 3: **Define every task as a cluster centroid.**

The number of task to be assigned will determine the number of clusters. Image 4.1 has 4 clusters because there are 4 tasks to be assigned and the cluster centroids are the locations assigned to those tasks in which agents are trying to reach.

Step 4: **Define formula to apply the number of free agents in each cluster.**

When agents enter a cluster, that means that they will compete with each other inside that cluster for that task. This competition is based upon their cost functions. Therefore, it is important to determine the number of agents inside each cluster and this is sensible to the ratio between the number of free agents and the number of tasks to be assigned:

(a) scenario in which the number of free agents and tasks are equal.

There always will be one agent inside each cluster.[See Image 4.4].

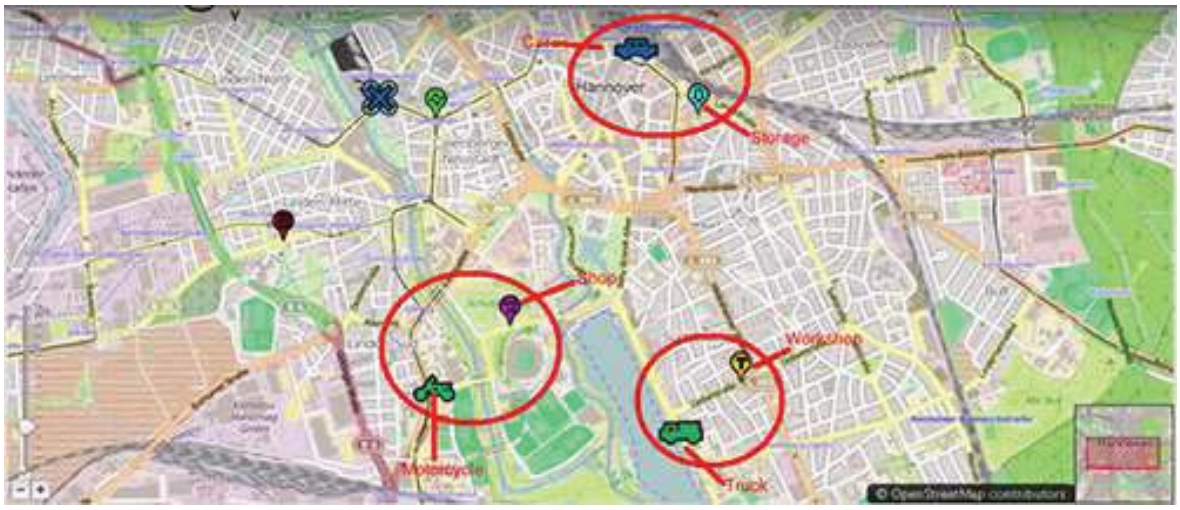

Figure 4.4: CTA same number of tasks and agents

(b) scenario with more tasks than agents

For this scenario, the extra tasks enter a queue and **4a** will be applied, consequently, there always will be one cluster for each task with one agent inside each cluster(see Image 4.4).

(c) scenario with more agents than tasks

The formula is given by C = N/T , where C means the number of candidates in each cluster; N equals to the total number of free agents; and T is the number of tasks to be assigned. In this scenario, the division between N and T will always be a integer value with a positive remainder. Image 4.3, three tasks need to be assigned and there are seven free agents competing for the tasks. The integer division between the number of agents and tasks equals to : 7/3 = 2 ; remainder = 1. This means that each cluster would have initially 2 agents inside their clusters.

The remainder value is stored in a variable and loops through the clusters in the priority established in **step 2**, adding a candidate to each cluster and decreasing the value of this variable by one at each iteration. The loop continues until the value of the variable remainder reaches zero. For the example above, the remainder is 1, therefore cluster 1 will receive one new candidate inside its cluster. If the remainder were 2, cluster 1 and 2 would have 3 candidates while cluster 3 would remain with 2 candidates.

Step 5: **Defining cluster aggregation rule**

After defining how many agents there are going to be in each cluster, it is important to determine the agents that are going to be inside each group. In order to determine which agent join a cluster, every agent will have their ratio between speed and distance to cluster centroid calculated. Therefore the algorithm begins an iteration process in which the first task in taken, all R values are calculated for that task, the algorithm picks

the N cheapest R values to the cluster. N is the number of candidates calculated in **step 4** for this task and R = D/S ; where R means the ratio between euclidean distance(from vehicle to centroid) and vehicle speed. This implies that the ratio between every free agent is calculated with respect to the current cluster centroid. In case R values are equal, agents properties are taken into account as a tie breaker. For instance, in the contest study case, the tie breaker could be the amount of gasoline the vehicle has or the amount of weight the vehicle is carrying. It is important to point out that Image 4.3 can only be drawn after the end of step 6. For this reason, in order to figure out that the agents assigned to cluster 1 in Image 4.3 are: agent3, agent4 and agent7, the ratio between agents 1,2,5,6 to task1 also had to be calculated. However, the number of ratio calculations are decreased as agents join clusters. For example, after agents 3,4,7 join cluster 1, only the ratio between agents 1,2,5,6 and task 2 are required to determine the agents to join cluster 2. The cluster aggregation process repeats until all clusters are fill out with all free agents.

Step 6: **Defining candidate to assign task inside each cluster**

In order to determine the agent to execute a task, each candidate has to call the score function in regard to its centroid. The agent who returns the highest value in the score function will be assigned the task.

Why not assign the agent with the highest ratio(step 6) directly to a task without calling any score function?

For the contest study case, the land route distance to a goal is never the same as the distance in a straight line. Therefore, even if a vehicle is faster and closer in straight line to a goal, it may take longer to reach the goal than other another vehicle. This rule could be applied if all vehicles are drones and vehicles never need to dodge obstacles, in this case the algorithm step 6 would be the score function itself. However, this is a very unrealistic scenario. Therefore, the step 7 is necessary for most realistic scenarios.

What happen to agents not assigned tasks inside clusters?

Not chosen agents remain idle waiting for a new task to be assigned. In order to be assigned a task, an agent has to score the best value in the objective function in comparison to its candidates. The agent speed and proximity to goal will increase the vehicle probability to be assigned a task. If the frequency of new tasks is very high, scenarios in which there is a single task to execute for a single agent will happen frequently, and even if an agent scores a low value in the objective function, the task will be assigned to him because there won't be any competition.

4.4 Comparison Contract Net Approach - Extended Implementation

The standard CNET implementation doesn't allow multiple agent bid at once and neither specify how an agent choose one task to bid. Therefore, similarly to most approaches that tries to improve the CNET, this restriction will be taken out. The CNET implementation will follow the rules:
All agents will bid for the first task, the agent with the highest score in the objective function will be assigned the first task, for the following tasks, this agent is excluded and the rest of agents compete for the remaining tasks. This process repeats until all agents are assigned tasks. It is important to point out that this strategy won't guarantee the optimal set of assignments as described in 2.2.1 and will increase the algorithm complexity from lineal to polynomial. However, it is good strategy to balance between speed and solution quality.

Image 4.3, there are 3 tasks to be assigned to 7 agents. In the contract net extended approach, for the first task, 7 agents will bid for the first task. For the second task, 6 agents will bid. Lastly, for the third task, 5 agents will bid for the last task. Resulting in 18 score function calls. If we look at the picture, some score function looks like a waste of time, because some agents are clearly too far away from these tasks, so they are clearly not the best solutions. In the proposed approach, the heuristics will cluster candidates and the score function will be called only for candidates to 'cluster centroids'. Consequently, for the task 1, only two score function will be called, for the second task, three calls and task for the third, two calls. As a consequence, the score function will be called just 7 seven times instead of 18. The idea is similar to [20], that is a approach in which each auction is performed among neighboring groups of agents and requires only local communication. In other words, it tries to reduce the number of calls to the score function based on a heuristic.

4.5 Algorithm complexity

This approach could dramatically decrease the number of score function calls and would work as a 'converged local search' because only the best candidates for each task will compete with each other. Candidates that are extremely far away from a goal are automatically excluded by the heuristic, resulting in less processing waste. The algorithm complexity is lineal, that is: one score function call to each one of the agents looking for a task. For the image 4.2 example; the number of score functions calls would be 6, because there are 6 agents attempting to get a task. The solutions quality would be very similar to the CNP if the scenario has few obstacles or if the land route from peer-to-peer is not very different from the distance in straight line. In other words, the more irregular is the land layout, the more the solutions will decrease in quality.

4.6 Dynamic Task assignment

It is important to keep in mind that the cluster layout will change according to the task assignment demand. For example, let us suppose that the tasks in figure 4.2 have to be assigned in instant of time 1, the algorithm runs and assigns the task 1 to agent 1, task 2 to agent 3 and task 3 to agent 5. As a consequence, agents: 2, 4 ,6 and 7 will remain idle until new tasks come up. Few seconds later(time step 10), two new tasks come up and the other agents previously assigned haven't finished their tasks yet. Therefore, agents : 2, 4 ,6 and 7 will compete for the new two tasks as described in figure 4.3.

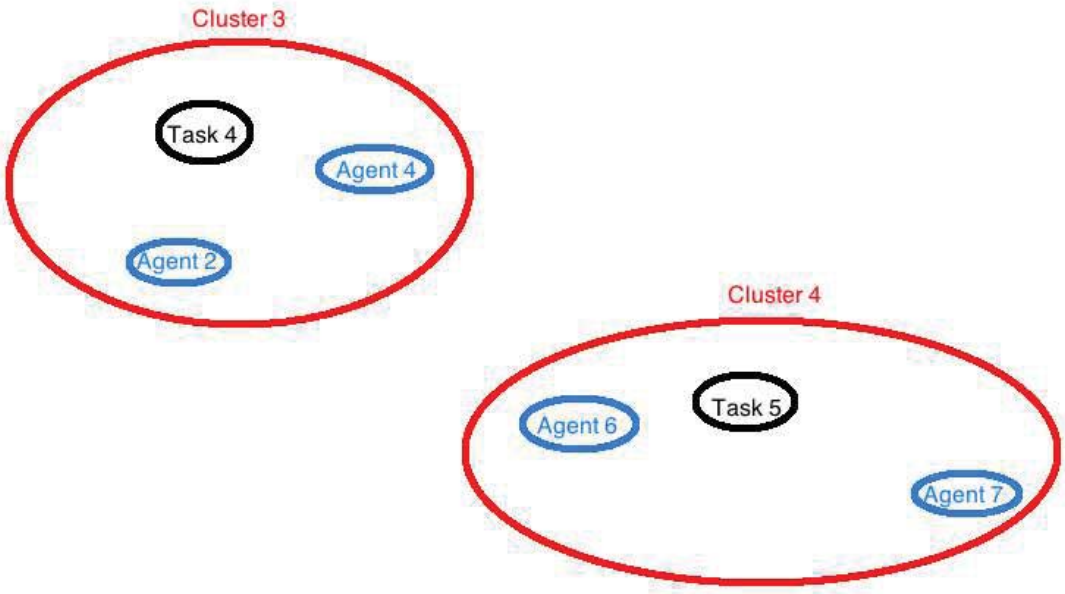

Figure 4.5: Clustering Task Assignment - time step 10

It is important to notice that images 4.2 and 4.3 are not printed in the same scale , but the positions of agent 2, 4 , 6 and 7 are the same in both pictures since they weren't assigned any task , they had to remain still. Therefore, let us suppose that agents 2 and 6 are assigned the tasks three and four, consequently: four tasks would be undergoing in total, four agents would be working and two idle. Latter on, when more tasks appear, new clusters will be created to determine task assignment. In addition, the agents who finish their tasks can compete for new task because only free agents do it. In conclusion, for every new assignment routine, new clusters are generated with free agents in order to find out the most suitable candidates.

4.7 Suitable scenarios to apply the approach

The main goal is to write an efficient algorithm to address task assignment issues in the field of transport logistics. This algorithm is also suitable for any time critical task assignment problem where agents have to move from one place to another and when the score function is very costly. The CTA approach should be able to find solution much faster than the CNP and provide solutions of similar quality because it is going to call the score function less times. In addition, the solution design is much simpler due to the fact that it doesn't implement a complex protocol like the CNP. The process of defining best candidates to a job lies in a heuristics instead of a bidding operation. In comparison to the ILP, it also should converge faster, because the clusters will work as converged solutions, in other words, the candidates to each task will always return high values in the score function. In comparison to Field-Based, this approach is only relevant when peer-distance in not the the only factor to determine task assignment. If the scenario involves only flying vehicles that never need to dodge obstacles , the FBTA is more applicable than the Clustering Task Assignment approach because the goal of choosing the best candidates lies in the fact that not always the closest and fastest agent is the most suitable agent for the task. For example, vehicle 1 could be 50 km away in a straight line to a goal, however the total land route path equals to 100km. While the vehicle 2 could be 60 km away in straight line to a goal but his ground route to the goal equals to 65 km. Therefore even if the vehicle 1 is faster and closer in straight line to the goal, the vehicle 2 will outrun him.

For any other kind of problem where time is not critical and score function is not costly, the CNP should perform better in terms of solution quality, because it doesn't implement any heuristic. In other words, CNP should provide a better set of solutions if its has more processing time.

4.8 Sample of scenario in which the approach could be useful

Let us suppose that we have aerials robots that need to move around an environment, the world is complex and robots have to dodge obstacle in order to reach their goals. The objective function is given by the cost to move from one place to another. This cost can be broken down into battery consumption and time taken to avoid obstacle. In this scenario agents share surrounding information with each other. Since agents share their locations with each other and they have access to shared goals, this approach could be used to determine which agent to send to desired location.

4.9 Scenario in which CTA will perform poorly

Since the algorithm goal is to sacrifice solution quality in order to get an answer faster, sometimes the solutions provided may not be very good.

Image 4.6, describes an scenario in which the heuristic will assign task 3 to a not very

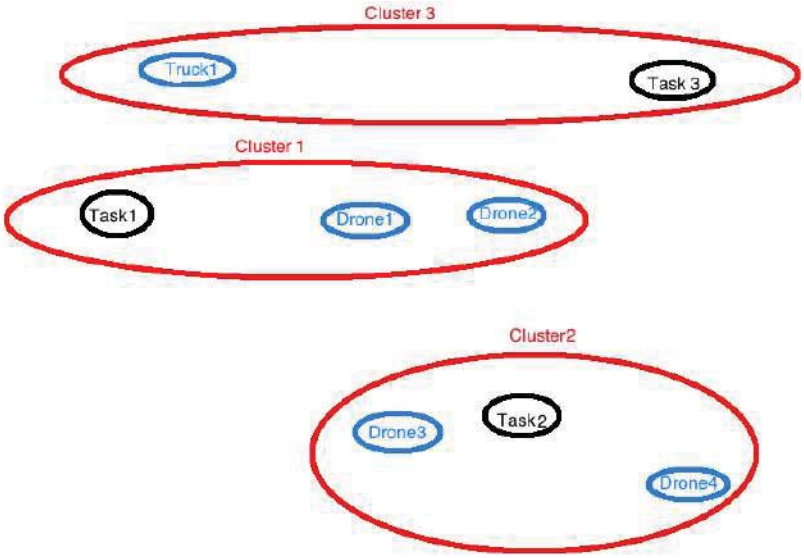

Figure 4.6: Scenario in which CTA might not perform well

suitable agent. In this scenario, at the end of the algorithm, task 1 will be assigned to drone1; task 2 to drone3. However, task 3 will be assigned to truck1 while drone2 and drone4 would remain idle. In this scenario assigning drone2 to task 3 would be much better because drone2 is faster and closer to its goal. Since the heuristics established drone1 and drone2 as task 1 candidates, even when one candidate is chosen to execute a job, the remaining candidates can not enter different clusters and thus competing for different tasks. In other words, when an agent looses the score competition, it immediately remains idle waiting for a new task. This is the worse case scenario, however the heuristic is still applies its main goal: reduce the time taken to assign task.

Chapter 5

Applications of CTA to the 2016 Multi-Agent Programming Contest

5.1 Scenario description

In this chapter we are going to analyze the whole process of task assignment using the CTA algorithm applied to the example in Image 4.3. The explanations here follow the steps present in 4.3 - Algorithm steps. This scenario represents the instant of time zero of one Multi-Agent Programming Contest simulation. In our scenario, we have:
*7 free agents
*3 tasks to be assigned
The agents are vehicles(their properties are described in table 5.1) who need to buy items at shops. Therefore, the tasks(properties described in table 5.2) are the three locations in which agents want to reach in order to buy items. In addition, theses tasks have no priority over each other. Let us suppose that Image 4.3 is represented inside a cartesian coordinate system whose coordinates range from 0 to 10 in X and Y axis.

Table 5.1: Vehicle properties - (Coordinates are not perfectly scaled)

Vehicle Id	Vehicle Type	Speed	X coordinate	Y coordinate
Agent 1	Truck	1	3	9
Agent 2	Drone	5	1	7
Agent 3	Car	3	7	7
Agent 4	Motorcycle	4	10	2
Agent 5	Car	3	4	3
Agent 6	Car	3	3	4
Agent 7	Drone	5	7	3

Table 5.2: Task properties - (Coordinates are not perfectly scaled)

Task Id	Location name	X coordinate	Y coordinate	Creation Date
Task 1	Shop1	9	5	2017-07-17 10:00:00
Task 2	Shop2	2	8	2017-07-17 10:00:01
Task 3	Shop3	2	2	2017-07-17 10:00:02

5.2 Defining tasks priorities

As mentioned in the scenario description, the provided tasks have no priorities, therefore they are sorted by creation time. The list is sorted in decreasing order, meaning that the oldest tasks are the first ones to enter in queue, following a FIFO(First in First Out) queue policy. As as consequence, the sorted list is the following: task1 , task 2 and task3.

5.3 Define every task as a cluster centroid

The first thing to do is to assign: task1, task2 and taske 3 as cluster centroids. So that, latter on free agents can join them.

5.4 Define formula to apply the number of free agents in each cluster

Now we need to determine which case(4.3 - step 4) we will use to assign free agents to tasks:

- number of agents and tasks are equal
- there are more tasks than agents
- **there are more agents than tasks**

In this case, the formula in **4.3 - step 4c** applies, because there are seven agents and three tasks. Therefore, the formula: C = N/T , where C means the number of candidates in each cluster; N equals to the total number of free agents; and T is the number of tasks to be assigned. Replacing the variables by values, we have: C = 7/3 = 2 ; since we want the integer division; an 7.33 result would be wrong. Initially each one of the tasks will have two candidates. Since this division has a remainder, task 1 (which is the first cluster according to create time) will be assigned the extra candidate. In the end, task 1 will have 3 candidates while cluster 2 and 3 remain with two candidates.(See Image 4.3).

5.5 Defining cluster aggregation rule

This process begin with a loop. The algorithm has to iterate over all cluster according to their priority list(4.3 - step 2). In our example cluster 1 with go first, then cluster 2 and lastly cluster 3. In order to determine which agent join a cluster, every agent will have their ratio between speed and distance to cluster centroid calculated. Theses values are stored inside an array and the lowest values according to the number of candidates that will join the cluster. Cluster 1 has to has three agents inside. However all free agents ratio have to be calculated in order to determine the three best who join the cluster.

The ratio of each one of these free agents are calculated in table 5.3:

For the example, Agent3(car), Agent4(motorcycle) and Agent3(drone) would be assigned as cluster candidates because they have the three lowest ratio values(in bold). The remaining

Table 5.3: Ratio calculation between Cluster 1 and free agents

Vehicle Id	Vehicle Type	Speed	Euclidean Distance to Shop 1	Ratio
Agent 1	Truck	1	7.21	7.21
Agent 2	Drone	5	8.24	1.65
Agent 3	Car	3	2.82	**0.94**
Agent 4	Motorcycle	4	3.16	**0.80**
Agent 5	Car	3	5.38	1.80
Agent 6	Car	3	6.08	2.03
Agent 7	Drone	5	2.82	**0.56**

agents are free to be assigned to other clusters. The algorithm keeps iterating until all clusters are filled out with all candidates.

5.6 Defining the candidate to assign task

In order to determine the agent to execute a task, each one of the candidates have to run their score function, and the candidate with the best score will be assigned the task. In this case, vehicles in the first cluster return the values bellow in the score function, this function returns an float value according to (4.2 - Cost function complexity - study case) , the lower the value of f(x), the better. The objective function is calculated by:
f(x) = W1(S) + W2(G) + W3(I) ; where S means the numbers of steps taken by the vehicle to arrive to the destination. G is the amount of gasoline consumed to arrive to the destination and I represents the item price to be purchased. W1,W2 and W3 describe the weight given to this set of criterion. These values can vary according to the way the objective function was modeled, in my implementation these values are: W1 = 1 ; W2 = 0.2 ; W3 = 0.1. After calculating the route between vehicle and destination, we can find the values of S and G. The values provided by I is returned by the server when agents get close to shops. Therefore, replacing this values into the variables, we have the following values in the objective function:

- Agent3(car) score function in respect with Shop1:
  f(x) = 1(15) + 0.2(100) + 0.1(150) = **50**
- Agent4(motorcycle) score function in respect with Shop1:
  f(x) = 1(10) + 0.2(80) + 0.1(130) = **39**
- Agent7(drone) score function in respect with Shop1:
  f(x) = 1(5) + 0.2(50) + 0.1(100) = **25**

According to the score function, Agent7(Drone) would be assigned Task1(Shop1). Agent3 and Agent4 will remain idle waiting for a new task to be assigned. The same procedure is executed for cluster 2 and cluster. In the end, 3 agents are assigned tasks and the 4 remaining agents remain idle.

Chapter 6

Results

In order to analyze the proposed algorithm performance, some comparison with the Contract Net Protocol will be made, because this protocol is widely known across the literature. In addition, the CNP is often used to assign tasks to problems like:

- **Routing**, e.g. model vehicle flow in transport networks..
- **Scheduling** e.g. flow shop scheduling where the resource management entity ensures local optimization and cooperation for global and local consistency.
- **Logistics** e.g supplier needs to collect primer matter in order to manufacture product and deliver to distributed clients

This protocol was also used by the Multi-Agent Contest 2016 [7] winner which is a task assignment logistic problem. Therefore, a comparison with the CNP would be a very good metric of comparison.

6.1 How CNET was implemented

The CNET version used in this experiment follows all the rules explained in 2.2.1. However with the extensions provided in 4.4. The complexity of this implementation is polynomial, like [12], explained in section 2.2.1. A comparison between CTA and CNEP standard could also be made, however the solutions provided by CNEP standard would be so bad, that this comparison would be irrelevant. It would happen because the standard CNEP doesn't specify an heuristic to define which task agents bid, therefore agents would need to bid random tasks, resulting in insignificant statistic data.

6.2 Metrics of comparison

6.3 Time Metric of comparison

To provide a good comparison, the time to assign a task will not be taken into account, because it is very variant to computer configuration. Moreover, this metric can be very volatile to the score function implementation. In other words, it could vary too much depending on how optimized the score function is. For example, one person could model one score function more efficiently than other. Or one person could wright a more efficient piece of

code to implement the same thing rather than another person. In the end, results can be very different if two developers try to solve the same problem with same strategies. Therefore, the only parameter to be taken into will be the number of calls to the score function because this metric is more feasible to compare heuristic algorithms efficiency. In addition, this metric is better to compare solution design efficiency, differently from CPU clock cycles which is very sensible to software development implementation.

6.4 Performance Metric of comparison

In order to contrast the algorithm efficiency, each task has to be evaluated according to its efficacy. Therefore, the tasks have to be measured according to an objective function, these parameters are: number of steps to arrive to the destination, amount of gasoline consumed, amount of money required to execute some action(see 5.6). However the only performance parameter that will be taken into account in this comparison will be the number of steps to arrive to the destination, because it has more relevance rather the other parameters. Therefore, the task are measured in terms of the number of steps to arrive at a destination. In the contest, this depends on the vehicle's speed and the distance to be traveled. If a vehicle has to pull up to recharge, some extra steps are also taken into account. In the simulation, each step equals to one second, therefore 10 steps to arrive at a destination, would also mean 10 seconds to arrive. For a more realistic comparison, the average steps of all assigned agent will be compared. Therefore, some agents could be assigned more efficient task rather than other from algorithm to algorithm. However, the goal here is to evaluate the average of steps assigned from both algorithms and contrast them in the end.

6.5 Scenarios of comparison

6.5.1 Single run scenario:

*16 agents have to be assigned 16 tasks.

The agent are instantiated randomly in the map, 16 random facilities are chosen randomly as destinations. Firstly, a single assignment will run for both algorithm with agents and destination initially setup equally. The main goal is to analyze if decisions are going to be similar. If decision are very different the goals is to understand why. The number of 16 agents and tasks are chosen for the single run scenario because this is the most complex task assignment scenario possible in the Multi-Agent Contest 2016. Two tables will be provided for this comparison, the first one shows: destinations, agents assigned, steps to arrive(solution quality) and number of calls to the score function. The second table displays the differences(Δ) between both algorithms assignments. Positive values mean that CTA outperform CNET.

6.5.2 One hundred runs scenario:

*In the one hundred times execution scenario, each run will have a different numbers of agents and facilities. The range of values vary between 1 and 16. The goal is to extract collective statistics of hundreds of runs in order to find out which algorithm takes less time to assign tasks and how good are the solutions provided. Different from the single run scenario, the goal here is to determine how both algorithms behave in a large range of different scenarios. At the end of 100 executions, the average score functions calls, total steps assigned to all agents will be output and the number of runs in which the set of solutions provided by Contract Net outperforms CTA is calculated.

6.6 Statistics

Table 6.1: Single run - Contract Net

Destination	Assigned Agent	Steps to Arrive	Score Function Calls
Charging station 1	Drone 2	6	16
Charging station 3	Drone 4	7	15
Charging station 7	Motorcycle 3	8	14
Charging station 4	Drone 3	3	13
Charging station 5	Motorcycle 2	6	12
Charging station 2	Car 3	7	11
Charging station 0	Drone 1	3	10
Charging station 8	Motorcycle 1	18	9
Charging station 6	Motorcycle 4	7	8
Shop 4	Car 1	28	7
Shop 5	Car 4	34	6
Shop 1	Truck 3	54	5
Shop 0	Car 2	74	4
Shop 6	Truck 1	118	3
Shop 2	Truck 4	104	2
Shop 3	Truck 2	198	1
Total Score Function Calls	Total steps all tasks		
136	675		

Table 6.2: Single run - CTA

Destination	Assigned Agent	Steps to Arrive	Score Function Calls
Charging station 1	Drone2	6	1
Charging station 3	Drone4	7	1
Charging station 7	Motorcycle3	8	1
Charging station 4	Drone3	3	1
Charging station 5	Car 1	9	1
Charging station 2	Car 3	7	1
Charging station 0	Drone 1	3	1
Charging station 8	Motorcycle1	18	1
Charging station 6	Motorcycle4	7	1
Shop 4	Motorcycle2	14	1
Shop 5	Car 4	34	1
Shop 1	Car 2	56	1
Shop 0	Truck 3	107	1
Shop 6	Truck 1	118	1
Shop 2	Truck 4	104	1
Shop 3	Truck 2	198	1
Total Score Function Calls	Total steps all tasks		
16	699		

Table 6.3: Assignment variation between CNET and CTA

Destination	Δ Assignments(CNET/CTA)	Δ Steps
Charging station 1	same agent	0
Charging station 3	same agent	0
Charging station 7	same agent	0
Charging station 4	same agent	0
Charging station 5	different agent	-3
Charging station 2	same agent	0
Charging station 0	same agent	0
Charging station 8	same agent	0
Charging station 6	same agent	0
Shop 4	different agent	+14
Shop 5	same agent	0
Shop 1	different agent	-2
Shop 0	different agent	-33
Shop 6	same agent	0
Shop 2	same agent	0
Shop 3	same agent	0
Δ Score Function Calls	Δ Total steps all tasks	
+120	-24	

Table 6.4: Performance and speed comparison CNET/CTA - one hundred runs

	Average number score function calls	average total steps assignment all tasks
CNET	31	84
CTA	7	108
	CTA 442% less score functions calls than CNET	CTA has solutions 28% better

Table 6.5: Solution quality comparison CNET/CTA - one hundred runs

	Number of times best solutions were found
CNET	13
CTA	26
Same	61

Chapter 7

Conclusions

In this project we have studied the definition and difficulties of task assignment problems. In order to understand these concepts, we have previewed classical task assignment problems present in the literature. Later on, we have understood what are time critical task problems which are the kind of problems addressed by this project.

In chapter 2, the most popular approaches to solve task assignment problems were introduced. Integer programming suffer the same as related search approach: getting stuck in local minimal. The bidding approaches can be very solution inefficient if the number of bids are limited to one in every iteration like in the standard CNET standard specification. Excluding this restriction can result in a optimization problem to determine the best set of assignments. The field based approaches can be effective if the only required parameter to determine assignment is the proximity to a task.

In chapter 3, we have seen how task assignment algorithms can be applied to solve logistics and supply chain problems. In addition, we have seen the Multi-Agent Programming Contest that organized by Clausthal University of Technology(Germany) every year. This contest was designed to motivate research in the field of multi-agent systems. The 2016 contest scenario requires time critical task assignment algorithm in order to write efficient solutions because requests have to be sent to the server every second, if requests time out, agents remain idle. Besides, the applications developed for the 2016 contest could also be applied to logistics and supply chain problems because in this scenario: vehicles move around the city collecting and delivering items in order to fulfill jobs. In other words, it simulates real logistics problems in which companies have to buy basic material to craft goods, so that it can sell items to clients later on in different locations.

In chapter 4, we presented the Custering Task Assignment Approach(CTA) which is a time critical task assignment algorithm that tries to address the flaws of most popular approaches. The algorithm avoids the search approach in order to prevent getting stuck in local. In addition, it has lineal complexity like the standard CNEP approach. Besides, it uses an heuristic to determine which cluster each agent will join. This approach works similarly to an efficient heuristic applied to the CNEP to determine which task each agent will bid. Furthermore, the CTA takes into account vehicle speed and proximity to goal in order to determine which

agent will join each cluster.

The chapter 5, provides an example of the CTA applied to the 2016 Multi-Agent Programming Contest. It describes the whole assignment process steps by step instantiating agents and tasks with values.

In the results chapter, the CTTA is measured against CNEP in terms of assignment speed and solution efficiency. It explains why the CNEP was chosen as the comparison target. Besides, this chapter explains the metric of comparisons, scenarios of comparisons and attributes used to generate statistics.

7.1 Comparisons CTTA/CNEP

7.1.1 Comparison in terms of speed

In terms of speed CTT is much faster than CNEP extended version because it calls the score function much less. Table 6.1 and 6.2 show that CNEP required 120 more calls to the score function to determine assignments for 16 agents. Besides, for the contest study case, the algorithm that use CNEP has to have an optimized score function implementation in order to be able to determine agents action in less than a second. The reason why the gap between the number of score function calls is so large due to the complexity of both algorithms. CTA is lineal while CNEP extended is polynomial.

7.1.2 Comparison in terms of solution quality

In terms of solution quality CTT is better than CNEP extended version. By coincidence, the single run (table6.1) shows CNEP with slightly better solution qualities. However, the group runs(table6.3) comparison points out that in 26 out 100 runs, CTT provide better solutions than CNEP extended. The average total steps statistics show that CTA solutions quality are equal in 61% of the runs. In 40% of the runs, results were different. Statistics show that when solutions found differ, quality is affected extremely. As a consequence of this divergence, the general results highlights that CTA provide 28% better solutions.

7.2 Should people always use CTT instead of CNET?

In order to find the optimal set of assignment in table 6.1 using a blind search(brute force approach), each one of the 16 agents would need to bid for each one the 16 tasks, resulting in 256 calls to the cost function(see 4.2). Then, we would need to test all the possible combinations of tasks assigned to each one of the agents in a objective function that tries to minimize the sum of total steps. In other words, testing 16! possibles combinations of values would result in 20.922.789.888.000 objective function calls.

In the way CNET extended was implemented the optimal solutions are not guaranteed. In the scenario of table 6.1, all agents bid for task1(Charging station 1), drone 2 returns the lowest value in the objective function(6 steps). Therefore, this agent is assigned this task

and stops bidding for the following tasks. However, this is could be not the best assignment, because drone 1 could return an lower value in the objective function for a different task. The heuristic applied by CTT finds a way to take into account agents feature like speed and proximity in order to find the best suitable agents without requiring too many calls to the score function.

The CTT should always be used instead of CNET extended in any kind of task assignment problem in which agents have to move from one place to another and when the score function is very costly provided that solution quality has been statistically proven better.

Is is important to point out that solutions found by CTA outperform CNET even if with use of an heuristic (which is designed to find near optimal solutions in less amount of time) .

7.3 Achieved goals

The heuristic successfully achieved its goal which was to accelerate algorithm speed. In regard to solution quality, the algorithm performed better than expected, because we were expecting solutions slightly worse than CNEP. It seems that the scenario described in 4.9 is very unlikely to happen, therefore the algorithm weak spot turned out irrelevant to the general results.

7.4 Skills acquired

This project was very useful to my personal learning. Above all, I could learn much about task assignment algorithms. Writing some piece of code for the 2016 Multi-Agent Programming Contest, I learned how complex are multi-agent algorithms are, specially in terms of coordination between agents which is the key to write intelligent multi-agent software implementation. Synchronization between client-server and parallel programming are other important factors that have to be taken into account in order to write efficient multi-agent algorithms. Parallelism between agent actions is a feature that increase the algorithm complexity, however it is required because agent have to take actions at the same time. The problem of this feature is the difficult of debugging wrong coordination between agents, therefore the use of specific debugging tools is very important.

I also could learn much about task assignment algorithms and how this techniques can be applied to real life problems like logistics and supply chain. In the end, I could leave my contribution to the scientific community with my algorithm CTA that can be very handy not only to time critical task assignment problems but also to task assignment problems in general.

Chapter 8

Future research

As described in 4.9, the algorithm has some flaws. Sometimes, an agent who is not selected in the competition inside its cluster, could be the more suitable to execute another task. Therefore, the set of solutions might not be very good. In order to address this flaw, a new version of the algorithm could be created. A feature allow agents that are not selected to join different clusters, thus becoming new candidates for other clusters. This feature will decrease the algorithm speed, however increasing solution quality.

This feature could be added to the algorithm as a boolean parameter that asks the user if he wants to allow agents to join new clusters. The sequence of images: 8.1,8.2 and 8.3 explain how this new feature would be implemented. In image 8.1 for the standard CTA implementation, only 5 calls to the score function would be required to assign drone3 to task1, drone3 to task2 and truck1 to task3. However, as explained in 4.9, assigning truck1 to task3 would be a bad choice. The new version of the algorithm would call the score function twice for task1 (drone1, drone2)(see image 8.1). In the next step, drone2 could join cluster2, resulting in three candidates to this cluster(see image 8.2). These three agents would call the objective function and drone3 would be assigned to cluster2. In step 3, drone2 and drone 4 would join cluster 3, because they lost the competition in cluster2. They would compete with truck1 for task3. After calling the score function three times, drone2 would be assigned to task3 which is the optimal set of assignments given this scenario. This approach would increase the number of calls to the score function from 5 to 8, however providing must better solution quality.

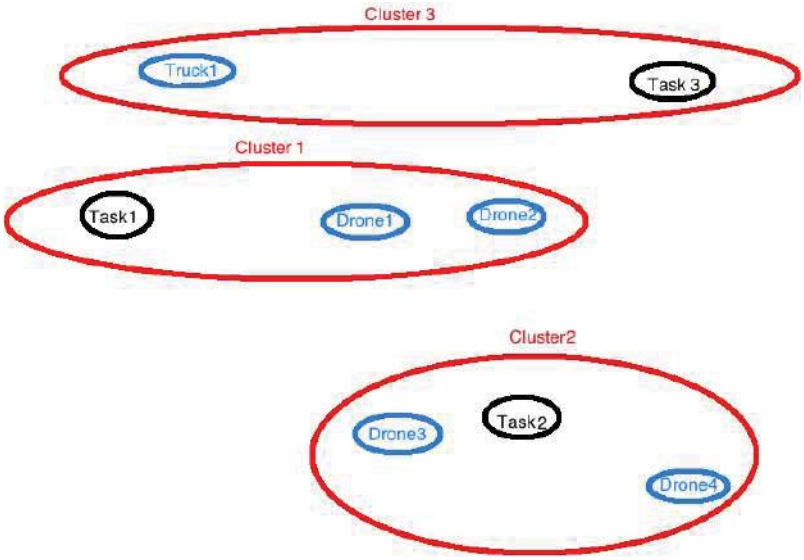

Figure 8.1: CTTA new feature - step 1

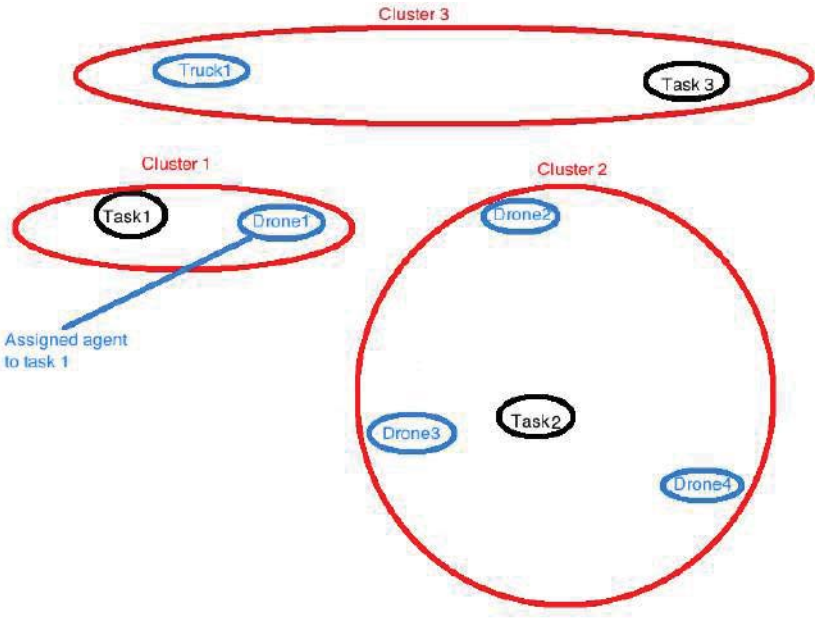

Figure 8.2: CTTA new feature - step 2

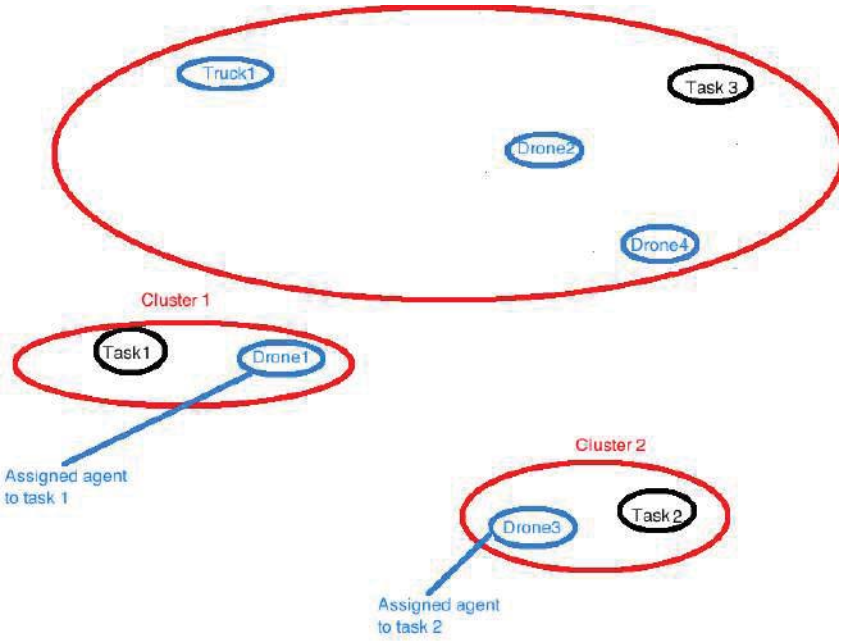

Figure 8.3: CTTA new feature - step 3

References

[1] HA Almohamad and Salih O Duffuaa. "A linear programming approach for the weighted graph matching problem". In: *IEEE Transactions on pattern analysis and machine intelligence* 15.5 (1993), pp. 522–525.

[2] J Randolph Andrews and Neville Hogan. "Impedance control as a framework for implementing obstacle avoidance in a manipulator". MA thesis. M. I. T., Dept. of Mechanical Engineering, 1983.

[3] Sigurd Angenent, Steven Haker, and Allen Tannenbaum. "Minimizing flows for the Monge–Kantorovich problem". In: *SIAM journal on mathematical analysis* 35.1 (2003), pp. 61–97.

[4] Reuven Cohen, Liran Katzir, and Danny Raz. "An efficient approximation for the generalized assignment problem". In: *Information Processing Letters* 100.4 (2006), pp. 162–166.

[5] Maria Pia Fanti, Agostino Marcello Mangini, and Walter Ukovich. "A distributed consensus algorithm for task allocation in supply chain management". In: *IFAC Proceedings Volumes* 45.6 (2012), pp. 566–571.

[6] Gerd Finke, Rainer E Burkard, and Franz Rendl. "Quadratic assignment problems". In: *North-Holland Mathematics Studies* 132 (1987), pp. 61–82.

[7] Artur Freitas et al. "Limitations and divergences in approaches for agent-oriented modelling and programming". In: *Engineering Multi-Agent Systems* (2016), p. 88.

[8] Lloyd Greenwald and Thomas Dean. "Solving time-critical decision-making problems with predictable computational demands". In: *Second International Conference on AI Planning Systems*. 1994, pp. 25–30.

[9] YY Haimes and Vira Chankong. *Multiobjective decision making—theory and methodology*. 1983.

[10] Meng Ji, Shun-ichi Azuma, and Magnus B Egerstedt. "Role-assignment in multi-agent coordination". In: (2006).

[11] Muhammad Kafil and Ishfaq Ahmad. "Optimal task assignment in heterogeneous distributed computing systems". In: *IEEE concurrency* 6.3 (1998), pp. 42–50.

[12] Anthony Karageorgos et al. "Agent-based optimisation of logistics and production planning". In: *Engineering Applications of Artificial Intelligence* 16.4 (2003), pp. 335–348.

[13] Oussama Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *The international journal of robotics research* 5.1 (1986), pp. 90–98.

[14] Bruce Krogh and Charles Thorpe. “Integrated path planning and dynamic steering control for autonomous vehicles”. In: *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*. Vol. 3. IEEE. 1986, pp. 1664–1669.

[15] Harold W Kuhn. “The Hungarian method for the assignment problem”. In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[16] Kristina Lerman et al. “Analysis of dynamic task allocation in multi-robot systems”. In: *The International Journal of Robotics Research* 25.3 (2006), pp. 225–241.

[17] Virginia Mary Lo. “Heuristic algorithms for task assignment in distributed systems”. In: *IEEE Transactions on computers* 37.11 (1988), pp. 1384–1397.

[18] David F Manlove et al. “Hard variants of stable marriage”. In: *Theoretical Computer Science* 276.1-2 (2002), pp. 261–279.

[19] *MASContest Multi-Agent Programming Contest: Scenario description*. `https://multiagentcontest.org/2016/scenario/`. Accessed: 2017-10-06.

[20] Nathan Michael et al. “Distributed multi-robot task assignment and formation control”. In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, pp. 128–133.

[21] Judea Pearl. “Heuristics: intelligent search strategies for computer problem solving”. In: (1984).

[22] Jay M Rosenberger et al. *The generalized weapon target assignment problem*. Tech. rep. TEXAS UNIV AT ARLINGTON, 2005.

[23] Senjuti Basu Roy et al. “Task assignment optimization in knowledge-intensive crowdsourcing”. In: *The VLDB Journal* 24.4 (2015), pp. 467–491.

[24] Omri Serlin. “Scheduling of time critical processes”. In: *Proceedings of the May 16-18, 1972, spring joint computer conference*. ACM. 1972, pp. 925–932.

[25] Reid G Smith. “The contract net protocol: High-level communication and control in a distributed problem solver”. In: *IEEE Transactions on computers* 12 (1980), pp. 1104–1113.

[26] Stephen L Smith and Francesco Bullo. “Target assignment for robotic networks: Asymptotic performance under limited communication”. In: *American Control Conference, 2007. ACC’07*. IEEE. 2007, pp. 1155–1160.

[27] Vidosav Stojanović. “M. Ben-Ari Principles of concurrent and distributed programming, 2/e Soft cover pp. 361, plus XV Addison Wesley, Harlow England, 2006 ISBN 0-321-31283-X”. In: *Facta universitatis-series: Electronics and Energetics* 19.2 (2006), pp. 334–336.

[28] Craig A Tovey. “A simplified NP-complete satisfiability problem”. In: *Discrete Applied Mathematics* 8.1 (1984), pp. 85–89.

[29] Deo Vidyarthi et al. *Scheduling in distributed computing systems: Analysis, design and models*. Springer Science & Business Media, 2008.

[30] Danny Weyns, Nelis Boucké, and Tom Holvoet. "A field-based versus a protocol-based approach for adaptive task assignment". In: *Autonomous Agents and Multi-Agent Systems* 17.2 (2008), pp. 288–319.

[31] Michael M Zavlanos and George J Pappas. "A dynamical systems approach to weighted graph matching". In: *Automatica* 44.11 (2008), pp. 2817–2824.

[32] Michael M Zavlanos and George J Pappas. "Dynamic assignment in distributed motion planning with local coordination". In: *IEEE Transactions on Robotics* 24.1 (2008), pp. 232–242.