2019 6th International Conference on Control, Decision and
Information Technologies (CoDIT'19) | Paris, France / April 23-26, 2019

The Multi-Armed Bandit Problem under Delayed Rewards Conditions in Digital Campaign Management

M. Martín[1], A. Jiménez-Martín[1], A. Mateos[1]

***Abstract*— In this paper, we account for a digital marketing content recommendation system, called *campaign management*, used by marketers to create specific digital content that can be issued or configured for viewing by certain population segments according to a series of business variables, user profile or behavior. We analyze the most representative allocation strategies to deal with the multi-armed bandit problem in a context with delayed rewards by means of a numerical study based on a discrete event simulation. Both batch mode and online update architectures are considered for feedback from the different contents displayed to users.**

I. INTRODUCTION

There are a wide variety of services in digital marketing used to offer personalized content to customers using the web, smartphones, social networks or email, such as *targeted advertisement*, *content recommendation*, *campaign management*, and *A/B test*.

Campaign management solutions help marketers to create specific content (web content, banners, emails, content on social networks) that can be issued or configured to be viewed by certain segments configured according to a series of business variables, user profile or behavior.

However, a customer may set up and run several campaigns at once. Therefore, conflict management must also be configured in a similar manner to rule production in expert systems. There are several configuration modes depending on the manufacturer or the system in question: the oldest campaign, the most restrictive, apply all campaigns, prioritize all campaigns, etc.

There are not many systems that apply online learning in these circumstances to identify the optimal configuration conflicts between two or more campaigns. This is a scenario where algorithms reported in the literature to solve the *multi-armed bandit (MAB) problem* are clearly applicable.

The name *bandit* stems from the image of a gambler playing with K slot machines. The gambler can pull the arm of any of the machines, which produces a reward payoff. Since the reward distributions are initially unknown, the gambler must use exploratory actions to learn the utility of the individual arms. However, exploration has to be controlled since excessive exploration may lead to unnecessary losses.

A gambler learning the distributions of arm rewards can use all past information to decide about his next action. Therefore, a *policy* or *allocation strategy* is an algorithm that chooses the next arm to play based on the sequence of previous plays and resulting rewards. The goal is to maximize the sum of the rewards received or equivalently to minimize regret.

There are two families of bandit settings [7]. In the first, the distribution of X_{it} is assumed to belong to a family of probability distributions, whereas, in the second, the rewards are assumed to be bounded, and policies rely directly on the estimates of the expected rewards for each arm.

Almost all the policies or allocation strategies in the literature focus on the first family, and they can be separated into two distinct approaches [10]: the frequentist view and the Bayesian approach. In the *frequentist view*, the expected mean rewards for all arms are considered as unknown deterministic quantities and the goal of the algorithm is to reach the best parameter-dependent performance, whereas the parameter is drawn from a prior distribution instead of considering a deterministic unknown quantity in the *Bayesian perspective*. Bayesian performance is then defined as the average performance over all possible problem instances weighted by the prior on the parameters.

Another family of algorithms for solving bandit problems is so-called *Thompson sampling* (TS) [13], consisting of randomly drawing each arm according to its probability of being optimal.

A review of the most important allocation strategies belonging to the above bandit settings can be found in [11], and a numerical study on the basis of five complex and representative scenarios was performed in [12] to compare their performances.

Table I shows the allocation strategies, together with information about their type, references in which they were introduced and whether or not they perform stochastic arm selection.

The allocation strategies analyzed in [12] do not account for delayed rewards. However, the reward for choosing one or the other action in real situations, such as Internet marketing advertising, clinical trials or content recommenders, is usually received with a delay after the time at which the action was executed, giving the algorithm the chance to work in this period and pick the actions to be taken in subsequent iterations without having to update its strategy.

There are two main types of *campaign management* architectures when dealing with how to update the feedback of the different contents displayed to users: batch mode and online update architectures.

The effect of delayed feedback in the MAB problem has been studied in the literature within different online learning scenarios and different delay configurations, for both

[1]Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Campus de Montegancedo S/N, Boadilla del Monte, 28660, Spain miguel.martin@alumnos.upm.es,{antonio.jimenez, alfonso.mateos}@upm.es

978-1-7281-0521-5/19/$31.00 ©2019 European Union

TABLE I
ALLOCATION STRATEGY FEATURES

Method	Type	Reference	Is Stochastic arm selection performed?
UCB	Upper confidence bounds	Agrawal (1995) [1], Auer et al. (2002) [3]	No
DMED	Kullback-Leibler-based algorithms	Honda & Takemura (2010) [8]	No
KL-UCB	Kullback-Leibler-based algorithms	Garivier & Cappé (2011) [7]	No
KL-UCB+	Kullback-Leibler-based algorithms	Garivier & Cappé (2011) [7]	No
BESA	Non-parametric algorithm	Baransi et al. (2014) [4]	Yes
TS	Thompson sampling	Thompson (1933) [13]	Yes
PR1	Possibilistic reward	Martín et al. (2016) [11]	Yes
PR3	Possibilistic reward	Martín et al. (2018) [12]	Yes

adversarial ([2]) and stochastic feedback. A concise summary including the (expected) regret in the delayed settings is given in [9].

Considering a fixed and known delay, Dudik et al. [6] showed an additive penalty in the regret for the stochastic setting (with side information). The problem of delayed feedback has also been studied for Gaussian process bandit optimization [5].

Joulani et al. [9] provide black-box algorithms for delayed feedback in both the adversarial and the finite stochastic settings, assuming that there is a base allocation strategy (BASE) for solving the prediction problem without delay. The *queued partial monitoring with delayed feedback* (QPM-D) algorithm is proposed in [9] to deal with partial monitoring rather than bandit feedback for a finite stochastic setting.

In this paper, we conduct a numerical analysis to analyze the performance of the most important allocation strategies in the literature for the MAB problem in a scenario with delayed rewards in digital campaign management. To do this, we implement discrete event simulation to account for both an adaptation of the QPM-D algorithm proposed in [9] for the MAB problem and for the original MAB allocation strategies. We consider different reward distributions and stochastic delays, representing different digital campaign management scenarios within online and batch mode update architectures.

The paper is structured as follows. Section 2 introduces the proposed discrete event simulation to account for delayed rewards. Section 3 describes the numerical analysis carried out and the results. Finally, some conclusions are outlined in Section 4.

II. DISCRETE EVENT SIMULATION

In this paper, we propose applying discrete event simulation to both an adaptation of the QPM-D algorithm for the MAB problem and the original MAB allocation strategies.

We consider two types of events in the discrete event simulation: the reception of a delayed reward and the selection of an arm based on the base allocation strategy (BASE).

We use a list L whose elements represent pending events to be treated at specified future time instants. Each element in this list is represented by a tuple $(t\_event, t, reward, arm)$, where the possible event types are r (reward) and a (action), t is the time instant at which the event will happen, $reward$ is the delayed reward value and arm is the selected/pulled arm that produced the above reward. Note that, in the case of an action event, $reward = arm = 0$.

A FIFO buffer Q_i for each arm $i, i = \{1, ..., K\}$ is used to store delayed rewards; $a_{k_{last}}$ is a global variable with the last executed arm; $blackbox$ denotes whether or not the adaptation of the QPM-D algorithm for the MAB problem proposed in [9] is considered; and the $reward\_scenario$ and $delay\_scenario$ identify the reward distribution and delay type, respectively, both described in detail for the different campaign management scenarios analyzed in Section 4.

The system state consists of both the rewards stored in buffers Q_i and the base allocation strategy (BASE) in use. The system state remains unchanged until one of the following events occurs: the reception of a delayed reward, which is stored in the corresponding buffer Q_i, and the arm selection based on BASE, which involves a BASE update and leads to the emptying of buffers Q_i.

We consider a non-homogeneous Poisson process with intensity function $\lambda(t)$ to generate the time instants in which arms have to be selected/pulled and the respective delayed reward is generated.

In the main simulation routine, see Algorithm 1, we first create an empty FIFO buffer Q_i for each arm $i, i = \{1, ..., K\}$, generate the time instant at which an arm has to be pulled for the first time and select the arm to be pulled on the basis of BASE.

Next, we call the Arm_event function to treat this first event. Then, while the event list L is not empty, we check the event that will take place next and call the respective event function to process the event. The simulation ends when L is empty.

The argument in the Arm_event function is the time at which the event to be handled occurs, t_o (see Algorithm 2).

First, the number of trials (an arm is selected) is incremented. Applying the $blackbox$ adaptation of the base allocation strategy, we update BASE with the feedback from Q_k, including delayed rewards associated with the arm a_k, select a new arm a_k by the updated BASE and repeat the process until $Q_k = \emptyset$.

Applying the original allocation strategy, we just select the arm to be pulled on the basis of BASE. In this case, the current a_k corresponds to the arm selected at time instant t_o. Thus, $a_{k_{last}} = a_k$. The corresponding reward r_{t_o} is derived from the reward distribution taking into account the selected arm, and the delay τ is generated.

Algorithm 1 Main simulation routine

Data: (T = max. number of trials; K = no. of arms; a_k= arm $k, k = \{1, ..., K\}$, $a_{k_{last}}$: last selected arm; $blackbox$: whether or not the blackbox adaptation is used; BASE: base allocation strategy; $update\_architecture$: the update architecture, $reward\_scenario$: reward distribution; $delay\_scenario$: delay scenario)

$n_{trials} = 0$
for ($i = 1$ **to** K) $Q_i = \emptyset$
$t_{first\_arm} \sim NH - Poisson(\lambda(t), 0)$
$a_{k_{last}} \leftarrow$ BASE
Arm_event(t_{first_arm})
while ($L \neq \emptyset$) **do**
  Let $L_{min} = (t\_event_{min}, t_{min}, reward_{min}, arm_{min})$ **be**
  the next event in L
  if (t_event_{min} ==“r”) **then**
    Reward_event($reward_{min}, arm_{min}$)
  else
    Arm_event(t_{min})
  end if
end while
end

Algorithm 2 Arm_event(t_o)

Data: (t_o = time instant at which the event occurs)

$n_{trials} + +$
$a_k = a_{k_{last}}$
Next arm selection:
if ($blackbox$ ==TRUE) **then**
  while ($Q_k \neq \emptyset$) **do**
    BASE $\leftarrow Q_k$
    $a_k \leftarrow$ BASE
  end while
else
  $a_k \leftarrow$ BASE
end if
$a_{k_{last}} = a_k$
$r_{t_o} \sim getReward(a_{k_{last}})$
$\tau \sim getDelay(t_0)$

Store future events:
$L \leftarrow (“r”, t_o + \tau, r_{t_o}, a_{k_{last}})$
$t_{next\_arm} \sim NH - Poisson(\lambda(t), t_o)$
if ($n_{trials} < T$) **then**
  $L \leftarrow (“a”, t_o + t_{next\_arm}, 0, 0)$
end if
end

Finally, we store the future events in L. On the one hand, we store the event corresponding to the time instant at which the generated delayed reward will be received. Additionally, we have to store the event corresponding to the time instant at which a new arm selection is required. The new arm selection event will be stored in L only if the number of already executed trials (arm selections) is lower than T.

The arguments in the $Reward\_event$ function are the delayed reward value r and the respective pulled arm k. Applying the $blackbox$ option, we add the delayed reward r to the buffer Q_k. Otherwise, we update the base allocation strategy with the delayed reward value r.

The $getDelay(t_0)$ function in Algorithm 2 can be used to account for different delay scenarios. In some cases, it outputs a fixed value C throughout the algorithm iterations. However, this delay behaves stochastically in most real situations and has a different value for each time instant at which an arm is executed.

The allocation strategies shown in Table I will be used as BASE allocation strategies in the discrete event simulation algorithm to deal with delayed rewards. Their blackbox adaptations will be denoted by BB-strategy_name.

III. NUMERICAL ANALYSIS

In this section, a numerical analysis will be carried out in campaign management scenarios with both batch mode and online update architectures. The arms to be selected/executed correspond to the different campaigns that can be offered to the customers.

We analyze the MAB algorithms in Table I together with the adaptation of the QPM-D algorithm for the MAB problem which are used as the BASE algorithm.

In our digital campaign management scenario, we consider connections to a company homepage. The intensity functions shown in Fig. 1 represent a low ($\lambda_1(t)$) a high ($\lambda_2(t)$) and a very high ($\lambda_3(t)$) traffic situation, respectively, regarding the non-homogeneous Poisson process used to generate the time instants at which arms have to be executed, i.e. the time instants at which the customer accesses to the company homepage.

The three functions have a standard shape to account for web connections, including two time periods (in the morning and the afternoon) where most are concentrated. The intensity functions are measured in connections per second.

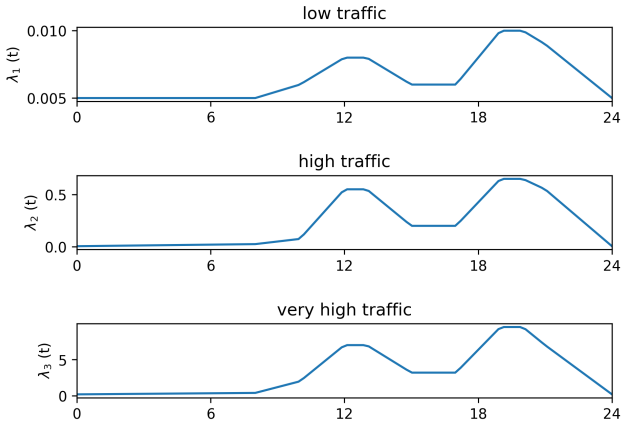

Fig. 1. Intensity functions for the web connections.

We account for two possible **reward scenarios** for each update architecture in the campaign management problem: a Bernoulli distribution with medium expected rewards, and a variant with rewards associated with the delays.

In the *first reward scenario*, the parameters are now very similar across the 10 campaigns and close to 0.5. We assume that success conditions are the same as in the first case, but customers are now well segmented enabling quite a targeted campaign. It is unreal to simulate success rates greater than 0.5, since even in a fine tune segmented campaign is very strange to find such high success rates are highly unlikely even in a campaign with very finely tuned segmentation.

We now use the following parameter vector for the ten arms/campaigns: $parameter = [0.5, 0.45, 0.45, 0.45, 0.45, 0.45, 0.45, 0.45, 0.45, 0.45]$.

Then, we simulate a scenario in the *second reward scenario* where the campaign content consists of a piece of news or information that should be carefully read by the user. The time that the user spends reading the news/information is a way of measuring the attention he/she pays to the content. Thus, the reward can be derived as follows:

- First, a Bernoulli distribution is used to model the success of clicking on the news. To do this, we again consider the parameter vector, $parameter$.
- If the user clicks on the news, the reward is then proportional to the time spent reading the item. A $[0, 480]$ secs. truncated exponential distribution is used to generate the reading durations. The reward will be equal to 1 if the user spends 480 seconds reading the news. Otherwise, the reward is normalized taking into account the respective amount of time. The parameter λ in the truncated exponential distribution is equal to 1/150, i.e., the average reading time is 150 seconds.

Different **delay scenarios** have also been taken into account in the *getDelay* function depending on the update architecture under consideration.

In an online update architecture, we consider one scenario with fixed delay values C, and an another scenario accounting for stochastic delays. If we consider a Bernoulli distribution with medium expected rewards, then the delay is randomly generated from a $[0, 300]$ truncated exponential distribution with parameter $\lambda = 1/80$. In the second reward scenario, we use a $[0, 480]$ truncated exponential distribution with parameter $\lambda = 1/150$.

When applying a batch mode update architecture, then the $getDelay$ function returns the difference between 24:00 and the time instant at which the function is executed. Thus, all rewards are received at 00:00.

The performance of the 16 MAB algorithms under consideration will be analyzed in all the scenarios in terms of their mean cumulative regrets and standard deviations.

The mean cumulative regrets are computed as follows: a K-armed bandit problem can be defined by random variables $X_{i,n}$ for $1 \leq i \leq K$ and $n \geq 1$, where each i is the index of one arm of a bandit and n refers to the round of play. Successive plays of arm i yield rewards $X_{i,1}, X_{i,2}, ...$, which are independent and identically distributed according to an unknown law with unknown expectation μ_i.

The *regret* of a given allocation strategy after n trials (plays) can be computed as

$$\mu^* n - \sum_{i=1}^{K} \mu_i E[n_i], \text{ where } \mu^* = \max_{1 \leq i \leq K} \{\mu_i\},$$

where $\mu^* n$ is the total reward that can be achieved given full knowledge of the problem. $E[\cdot]$ denotes expectation and n_i is the number of times arm i has been played by the allocation strategy during the first n plays.

Note that 1000 simulations of $T = 50,000$ trials (arm/campaign selections) are executed in the scenarios.

A. Campaign management with an online update architecture

We analyze the performance of the 16 allocation strategies under consideration for the six possible scenarios with an online update architecture and stochastic delays.

Table II shows the resulting mean cumulative regrets and standard deviation values in the reward scenarios concerning the Bernoulli distribution with medium expected rewards (Bern_M), whereas $\lambda_1(t)$, $\lambda_2(t)$ and $\lambda_3(t)$ are the intensity functions accounting for a low, high or very high traffic, respectively.

We randomly generate the delays from a $[0, 300]$ truncated exponential distribution with $\lambda = 1/80$ for reward scenario 1 (Bern_M).

Note that, the performance of the allocation strategy PR3 in the first reward scenario where a Bernoulli distribution is used for the reward distribution, is exactly the same as for PR2 and Thompson sampling (TS), as pointed out in [12].

Looking at the first three columns in Table II, corresponding to **reward scenario 1** (Bern_M), we again find that the performance of the original allocation strategies and their respective blackbox version is similar irrespective of the traffic and the mean cumulative regret is in some cases higher for the original allocation strategy although the opposite applies in others. The biggest difference in the mean cumulative regrets is 36.31 (8.4%) for BESA with high traffic. The standard deviations are similar for all the allocation strategies, except for BESA and BB-BESA, which are as much as six times greater.

In the **low traffic** scenario ($\lambda_1(t)$), PR2 and BB-PR2 outperform the other allocation strategies, followed by BB-PR1 (and PR1), BESA and BB-DMED. In the **high traffic** scenario ($\lambda_2(t)$), PR1 is the best allocation strategy, followed by BB-PR2, PR2 and BB-PR1, with similar performances, and further behind by BB-DMED and DMED. Finally, PR2 outperforms the other allocation strategies when traffic is very high ($\lambda_3(t)$), followed by BB-PR1, BBPR2 and PR1, with similar performances.

Thus, the original PR methods outperform the other allocation strategies in reward scenario 1 (Bern_M), irrespective of the traffic.

The last three columns in Table II, shows the resulting mean cumulative regrets and standard deviation values for

TABLE II
RESULTS IN ONLINE UPDATE ARCHITECTURE (BERN_L, BERN_V)

	Bern_M, $\lambda_1(t)$	Bern_M, $\lambda_2(t)$	Bern_M, $\lambda_3(t)$	Bern_V, $\lambda_1(t)$	Bern_V, $\lambda_2(t)$	Bern_V, $\lambda_3(t)$
UCB	742.84(150.95)	744.23(169.99)	780.8(151.75)	1155.66(90.58)	1187.47(111.35)	1213.45(109.57)
BB-UCB	715.41(159.65)	725.3(155.49)	753.36(159.42)	1153.47(90.06)	1180.69(130.58)	1131.56(110.77)
DMED	403.0(163.28)	417.76(177.24)	460.53(182.7)	570.33(181.24)	577.84(167.8)	659.01(176.5)
BB-DMED	378.86(163.0)	403.12(175.27)	426.64(171.06)	554.7(151.38)	574.32(158.79)	526.98(198.09)
KL-UCB	736.52(150.33)	725.76(157.51)	776.59(155.89)	909.56(109.66)	923.44(122.98)	949.47(114.16)
BB-KL-UCB	718.9(158.24)	727.74(150.06)	744.6(151.67)	906.81(108.16)	920.09(120.6)	699.26(126.9)
KL-UCB+	456.51(139.73)	456.24(144.95)	508.34(144.58)	624.65(127.73)	627.52(117.47)	644.45(120.14)
BB-KL-UCB+	453.29(139.26)	440.38(144.0)	465.53(126.22)	617.63(112.37)	609.73(123.61)	494.98(167.5)
BESA	401.83(575.82)	430.49(586.63)	430.14(574.81)	**376.86(489.9)**	**358.56(487.99)**	**305.72(350.99)**
BB-BESA	438.64(602.83)	457.85(594.14)	498.72(637.67)	1007.88(756.96)	487.9(608.52)	1087.91(808.51)
PR1	361.93(160.66)	**337.84(142.9)**	366.11(147.56)	708.87(130.97)	715.52(132.34)	738.45(132.06)
BB-PR1	346.51(152.91)	352.23(222.63)	357.8(156.98)	692.21(119.06)	701.49(120.05)	741.08(130.38)
PR2 (TS)	**338.63(148.39)**	349.8(156.31)	**341.4(146.07)**	448.18(115.41)	462.3(127.19)	506.55(120.86)
BB-PR2 (TS)	342.84(157.49)	347.96(144.64)	359.79(167.94)	432.4(98.41)	442.82(120.97)	470.29(158.54)
PR3				**345.86(115.2)**	**353.91(106.18)**	**396.41(113.98)**
BB-PR3				**335.58(102.12)**	**349.21(99.72)**	**349.97(120.68)**

120,000 trials in **reward scenario 2**, concerning the variant with rewards associated with the corresponding delays (Bern_V), where the delays are randomly generated from a $[0, 480]$ truncated exponential distribution with $\lambda = 1/150$.

BESA, PR3 and BB-PR3 are the best three allocation strategies, irrespective of the traffic under consideration.

Blackbox versions tend to outperform the original allocation strategies for **low traffic** ($\lambda_1(t)$), except for BESA, which is clearly better than BB-BESA.

BESA is the best allocation strategy in terms of the mean cumulative regret when the number of trials is lower than 83,700. For a higher number of trials, BB-PR3 is the best allocation strategy. PR3 is the second best strategy from trial 95,000 onwards. As the average number of trials per day is 500, which is equivalent to low traffic, BB-PR3 would outperform BESA from day 175 onwards. However, the usual duration of campaigns is from 5 to 20 days. In this case, BESA is definitely the best allocation strategy when traffic is low. However, BB-PP3 would be better in long-term campaigns.

It is also important to note that the standard deviation for BESA is about four times greater than for PR3 and BB-PR3.

In the **high traffic** scenario ($\lambda_2(t)$), blackbox versions tend to outperform the original allocation strategies without exceptions, although performances are very similar. BESA is the best allocation strategy in terms of the mean cumulative regret when the number of trials is lower than 105,000. For a higher number of trials, BB-PR3 is the best allocation strategy. PR3 is the second best strategy from trial 112,000 onwards. As the average number of trials per day is 20,000, which is equivalent to high traffic, BB-PR3 would outperform BESA from day 5.71 onwards. Thus, BB-PR3 is definitely the best allocation strategy when traffic is high.

Finally, blackbox versions clearly outperform the respective original strategies in the **very high traffic** scenario ($\lambda_3(t)$), except for BESA, which is clearly better than BB-BESA. BESA is the best allocation strategy when the number of trials is lower than 358,350. For a higher number of trials, BB-PR3 is the best allocation strategy. PR3 is the second-best strategy from trial 650,000 onwards. In high traffic, the average number of trials per day is 250,000. Therefore, BB-PR3 would outperform BESA from day 5.71 onwards. Thus, BB-PR3 is definitely the best allocation strategy when the traffic is high ($\lambda_3(t)$).

B. Batch mode update architecture

We now assume that the rewards associated with the actions performed by the users when the contents are displayed in the campaign management system are analyzed at 0:00 each day.

Table III shows the mean cumulative regrets and standard deviation values for the regret after 120,000 trials using the 16 allocation strategies under consideration for the six possible scenarios (two reward scenarios with low and high traffic respectively) with a batch mode update architecture.

Note that in scenario Bern_V, the reward value depends on the time the customer takes to read the corresponding campaign (piece of news or information). This reading time is randomly generated from a $[0, 480]$ truncated exponential ($\lambda = 1/180$).

BB-DMED outperforms DMED in the scenarios under consideration, whereas the original allocation strategies outperform the blackbox version of the BESA and PR methods in all scenarios. KL-UCB+ outperforms its blackbox extension in Bern_M scenario, whereas the opposite applies to Bern_V scenario.

In **reward scenario 1** (Bern_M), PR1 and PR2 are the best strategies with a very similar mean accumulated regret if traffic is low ($\lambda_1(t)$). They are followed by BESA. However, we find that BESA is the best allocation strategy for a number of trials lower than 29,630. As the average number of trials with high traffic is 20,000 per day, PR1 and PR2 outperform BESA from day 1.45 onwards. Thus, PR1 and PR2 are the best allocation strategies when traffic is low.

When traffic is high, PR2 is the best allocation strategy for any number of trials, followed by PR1 and BESA.

Finally, in **reward scenario 2** (Bern_V), BESA is the best strategy for both low and high traffic irrespective of

TABLE III

RESULTS IN BATCH MODE UPDATE ARCHITECTURE

	Bern_M, $\lambda_1(t)$	Bern_M, $\lambda_2(t)$	Bern_V, $\lambda_1(t)$	Bern_V, $\lambda_2(t)$
UCB	874.81(167.24)	1589.23(118.42)	599.41(58.81)	618.01(55.6)
BB-UCB	1249.5(1122.55)	1737.9(438.82)	400.76(317.49)	532.92(138.96)
DMED	1828.35(348.07)	2350.95(116.23)	698.75(55.66)	704.97(50.16)
BB-DMED	1109.94(1075.93)	1736.9(422.98)	343.4(310.11)	608.65(127.26)
KL-UCB	872.93(168.15)	1589.32(118.38)	541.98(68.51)	584.41(60.32)
BB-KL-UCB	1249.5(1122.55)	1737.9(438.82)	377.41(336.54)	531.55(128.41)
KL-UCB+	606.47(149.87)	1517.61(154.81)	446.31(72.74)	536.1(62.3)
BB-KL-UCB+	1249.5(1122.55)	1737.9(438.82)	414.84(333.58)	509.77(133.38)
BESA	390.69(436.57)	427.38(251.0)	**205.18(180.23)**	**229.4(152.62)**
BB-BESA	1369.22(1108.85)	1730.14(438.33)	360.19(335.46)	529.49(132.0)
PR1	**349.1(140.21)**	405.93(147.98)	452.57(78.48)	464.19(77.12)
BB-PR1	1362.33(1112.65)	1804.48(442.98)	499.37(321.03)	561.61(143.12)
PR2 (TS)	**351.06(155.5)**	**391.92(145.55)**	347.52(88.38)	352.61(90.7)
BB-PR2 (TS)	1413.95(1131.51)	1752.21(431.29)	410.44(337.29)	531.98(143.3)
PR3	-	-	288.15(86.72)	297.76(88.83)
BB-PR3	-	-	471.88(338.07)	538.77(143.96)

TABLE IV

BEST ALLOCATION STRATEGIES IN THE SCENARIOS UNDER CONSIDERATION

	Online update architecture		In batch mode update architecture	
	Bern_M	Bern_V	Bern_M	Bern_V
$\lambda_1(t)$	PR2	BESA/BB-PR3	PR1/PR2	BESA
$\lambda_2(t)$	PR1	BB-PR3	PR2	BESA
$\lambda_3(t)$	PR2	BB-PR3	-	-

the number of trials, followed by PR3 and further by PR2.

Thus, we can conclude that the BESA and PR methods are the best allocation strategies for campaign management with a batch mode update architecture.

IV. CONCLUSIONS

Table IV shows the best allocation strategies for the scenarios, accounting for different levels of traffic and reward distributions in online and in batch update architectures.

There is no one allocation strategy that outperforms the other in all scenarios under consideration. Four allocation strategies are the best in at least one of the 10 analyzed scenarios. PR2 is the best for four out of 10 scenarios, followed by BESA and BB-PR3 (three scenarios) and PR1 (two scenarios). In only two cases is the blackbox version of an allocation strategy the best option.

Thus, depending on the configuration of the campaign management under consideration, a MAB algorithm could be the most appropriate. In an online update architecture, if a Bernoulli distribution with medium expected rewards (Bern_M), PR2 is recommended with low or very high traffic, whereas PR1 should be used when traffic is high. Finally, BB-PR3 is the best algorithm for the variant with rewards associated with the delays (Bern_V) irrespective of the traffic.

In a batch mode update architecture, BESA and PR2 are the best algorithms for reward scenarios Bern_M and Bern_V irrespective of the traffic, respectively.

ACKNOWLEDGMENT

The research reported in this paper was supported by Spanish Ministry of Economy and Competitiveness project MTM2017-86875-C3-3-R.

REFERENCES

[1] Agrawal, R., 1995. Sample mean based index policies with O(log n) regret for the multi-armed bandit problem, *Advances in Applied Probability* 27, 1054-1078.

[2] Agarwal, A., Duchi, J., 2011. Distributed delayed stochastic optimization, *Advances in Neural Information Processing Systems* 24, 873-881.

[3] Auer, P., Cesa-Bianchi, N., Fischer, P., 2002. Finite-time analysis of the multiarmed bandit problem, *Machine Learning* 47, 235-256.

[4] Baransi, A., Maillard, O.A. Mannor, S., 2014, Sub-sampling for multi-armed bandits, *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, 115-131.

[5] Desautels, T., Krause, A., Burdick, J., 2014, Parallelizing exploration-exploitation tradeoffs with gaussian process bandit optimization, *Journal of Machine Learning Research* 15, 4053-4103.

[6] Dudik, D., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., Zhang, T., 2011. Efficient optimal learning for contextual bandits, *Proceedings of the 27th International Conference on Uncertainty in Artificial Intelligence*, 169-178.

[7] Garivier, A., Cappé, O., 2011. The KL-UCB Algorithm for bounded stochastic bandits and beyond, *Proceedings Conference on Learning Theory* 24, 359-376.

[8] Honda, J., Takemura, A., 2010. An asymptotically optimal bandit algorithm for bounded support models, *Proceedings of the 24th Annual Conference on Learning Theory*, 67-79.

[9] Joulani, P., György, A., Szepesvári, C., 2013. Online learning under delayed feedback, *Proceedings of the 30th International Conference on Machine Learning* 28(3), 1453-1461.

[10] Kaufmann, E., Cappé, O., Garivier, A., 2012. On Bayesian upper confidence bounds for bandit problems, *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 592-600.

[11] Martín, M., Jiménez-Martín, A., Mateos, A., 2016. Possibilistic reward method for the multi-armed bandit problem, *Proceedings of the 6th International Conference on Operations Research and Enterprise Systems*, 75-84.

[12] Martín, M., Jiménez-Martín, A., Mateos, A., 2018. The Possibilistic Reward Methods for the Multi-Armed Bandit Problem, *Neurocomputing* 310, 210-212.

[13] Thompson, W.R., 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples, *Biometrika* 25(3-4), 285-294.