Information Fusion 91 (2023) 644–664

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full length article

Deep reinforcement learning in agent-based simulations for optimal media planning

Víctor A. Vargas-Pérez [a,*], Pablo Mesejo [a], Manuel Chica [a,b], Oscar Cordón [a]

[a] Department of Computer Science and Artificial Intelligence (DECSAI), Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada (UGR), 18071 Granada, Spain

[b] School of Electrical Engineering and Computing, The University of Newcastle, Callaghan, NSW 2308, Australia

ARTICLE INFO

Keywords:
Deep reinforcement learning
Deep Q-Network
Agent-based modeling
Media planning
Marketing

ABSTRACT

Agent-based models establish a suitable simulation technique to recreate real complex systems, such as those approached in marketing. Reinforcement learning is about learning a behavior policy in order to maximize a long-term reward signal. In this work, we develop a deep reinforcement learning agent that represents a brand in an agent-based model of a market. The goal of the learning agent is to obtain a marketing investment strategy that improves the awareness of its corresponding brand in the marketing scenario. In opposition to conventional marketing investment strategies, the learned strategy is dynamic, so the agent makes its investment decision on-line based on the current state of the market. We choose the Double Deep Q-Network algorithm to train this agent on diverse instances of the model, each of them with different knowledge levels of the brand. First we adjust a subset of the hyperparameters of Double Deep Q-Network on two of the model instances, and then we use the best configuration found to train the agent on all the available instances. The brand agent learns a dynamic policy that optimizes brand's awareness levels. We perform an expert analysis of the policy obtained, where we observe that the learning brand agent tends to increase investment in media channels with greater awareness impact, but it also invests in other channels according to the situation and the characteristics of the model instance. These results show the benefits of having an on-line dynamic learning environment in a decision support system for media planning in marketing.

1. Introduction

Agent-based models (ABMs) form a powerful technique which is useful to understand, predict and control the behavior of complex systems [1]. ABMs allow us to represent complex systems through their most basic elements, using a population of individuals or agents with a predefined set of simple micro-rules. The interaction between these agents along with different simulations of the model will give rise to the emergent behavior or macro-effects of the system.

Media planning in marketing gives rise to a complex problem in which it is difficult to determine all the factors that affect the decision of each consumer, specially when the market involves the action of different competing brands and the situation of the environment changes at short intervals of time. Traditionally, marketers define the distribution of the investment budget over long periods of time such as one or two years. We call this a static media planning strategy. However, the rapid evolution of the market makes it desirable to define the distribution of the investment budget dynamically, i.e., in short time intervals.

The complexity of marketing scenarios makes agent-based modeling a suitable technique for building decision support systems for firms. An ABM can recreate a real market involving different consumers, business and media [2]. If the ABM is properly calibrated and validated with real data, it can be used to test the effect of different media planning policies on consumer behavior. This work considers and extends ABMs to design adaptive media planning policies that indicate the best action to apply according to the market situation at any time. Given the dynamic nature of the problem, we propose to combine agent-based modeling with reinforcement learning (RL) [3], a machine learning subfield that involves a set of methods to obtain action policies that maximize a long-term reward signal in a particular environment [4].

More precisely, our proposal is to use RL to train a learning agent that represents a brand in an agent-based marketing scenario. The aim of the brand agent is to learn a policy or media planning strategy that increases the awareness level of the corresponding brand as much as possible. The ABM used (extended from that introduced in [5]) has

* Corresponding author.
E-mail addresses: victorvp@ugr.es (V.A. Vargas-Pérez), pmesejo@decsai.ugr.es (P. Mesejo), manuelchica@ugr.es (M. Chica), ocordon@decsai.ugr.es (O. Cordón).

https://doi.org/10.1016/j.inffus.2022.10.029
Received 29 July 2022; Received in revised form 24 October 2022; Accepted 27 October 2022
Available online 9 November 2022

been calibrated using data extracted from real marketing scenarios, and involves a series of brands that try to maximize their awareness within a consumer network by making advertising investments in different media or touchpoints throughout the simulation time.

The contribution of this work is to integrate a well-known algorithm of deep RL such as Double Deep Q-Network (DDQN, [6]) to the context of a marketing and media planning ABM for some agents to react and learn from the changing conditions of the environment. Hence, a learning agent is trained to make decisions about the investment to be made in different advertising media based on the current state of the market. We believe that the use of deep RL is appropriate for this problem because it allows us to consider any state of the environment (the market) and to evaluate each possible action considering its long-term effect, which does not necessarily coincide with its immediate effect and is difficult to determine in a complex system.

In order to test the effectiveness and robustness of the proposed method, we will conduct an extensive experimentation in which a learning brand agent will be trained on a total of seven instances of the model. These instances show different dynamics of awareness evolution, defined by applying changes over two base instances obtained by calibrating the model with data from real marketing problems. This experimentation will involve an initial tuning of some of the hyperparameters of the learning algorithm. We will evaluate the dynamic media planning strategy learned by comparing its awareness results with those given by a static investment strategy previously followed by the real brand. Then, we will analyze the distribution of the budget made by the learning agent in each situation. Finally, we will provide a comparison of the method with a static strategy based on an econometric model, a common method used in the marketing industry and literature to make media planning decisions [7,8].

The rest of the paper is structured as follows. In Section 2 we present the background of this work related to agent-based modeling, RL, and the combination of both techniques. We describe in Section 3 the ABM for the marketing problem. Next, in Section 4 we formalize the RL problem addressed in this work, defining aspects such as the action space, the state space, and the reward signal. In Section 5 we show the results of the experimentation and compare those results obtained by the learning media agent with those given by the static baseline policy and the econometric model-based strategy. Finally, we show the conclusions of our work in Section 6.

2. Background

We detail in this section the background of the paper. Concretely, we show a brief overview of agent-based modeling in Section 2.1, explain theoretical aspects of RL necessary to understand the DDQN algorithm in Section 2.2, and explore previous works which combine both techniques in Section 2.3

2.1. Agent-based modeling

Agent-based modeling [1] is a simulation technique that allows us to take advantage of large amount of data to represent real complex systems properly, i.e, systems which are constantly evolving due to both external and internal factors, the latter as a result of the interaction of the individual elements of the system [9]. ABMs recreate these systems through the definition of a set of individuals or agents who are usually connected through an artificial social network [10]. In this way, the modeler must only define the characteristics and micro-rules that guide the behavior of the agents at individual level, without the need of making assumptions of the system behavior at the global level. Stochastic and discrete simulations of the model give rise to the macro-effects present in the real system.

This modeling technique offers better results than top-down simulation models when the system to recreate is complex and dynamic, as well as when the measure of interest is an emergent result of agent interactions [11]. This technique is also suitable for dealing with environments where the agents have heterogeneous properties and the decision processes of each agent are influenced by their individual properties, social interactions and seasonal components.

In order to properly build an ABM to address a problem of interest, the modeler must approach different design choices [11]. First, it is necessary to determine the part of interest of the complex system to be modeled to ignore noisy elements. Then, the modeler must define the types of agents together with their properties and behaviors. Another important aspect is the environment where agents operate, as it determines the influential external forces and the topology interaction of the agents. In relation to the objective and application of the model, the modeler has to define the input data of the ABM and the output measure to be collected (on a macro- and/or micro-basis). Finally, the temporal dynamics of the model must be established. This temporal dynamics means how the model and their variables are initialized at the beginning of the simulation, the meaning of every iteration step that will be repeated for the entire simulation time as well as this maximum time.

We find applications of this modeling technique in fields as diverse as ecology [12], sociology [13], epidemiology [14], political communication and terrorism [15], tourism [16], and marketing [2], among many others. In marketing, ABMs are useful to understand how consumers, brands and media in a market relate to each other. The simulation of these models also allows us to test the effect of micro-campaigns and marketing strategies (what-if scenarios) without the risk of testing them directly in the real world [2].

2.2. Reinforcement Learning

Reinforcement Learning (RL) [4] is a machine learning research field addressing problems where the goal is to learn what actions to take, given a state of a specific environment to maximize a reward signal. Normally, the taken action affects not only the immediate reward, but also the next state of the environment and consequently the future rewards.

These problems are commonly formalized as a Markov Decision Process (MDP), a 5-tuple (S, A, R, p, γ) where S is the set of possible states of the environment, A is the set of actions that the agent can perform, R is the set of possible immediate rewards that the agent can receive after applying an action, $p : S \times R \times S \times A \rightarrow [0, 1]$ is a function which represents the dynamics of the environment, and $\gamma \in [0, 1]$ is a discount factor which represents the preference of present rewards over future rewards. In this context, the agent interacts with the environment through a sequence of steps $t = 0, 1, 2, \ldots, T$ (being an episode a sequence of steps from an initial to a terminal state). On the other hand, a policy π is a function that assigns a probability distribution over the action space to each possible state of the environment. The objective of the agent is to learn a policy that maximizes the expected return $G_t = R_{t+1} + \sum_{k=1}^{T} \gamma^k R_{t+k+1} = R_{t+1} + \gamma G_{t+1}$, where $R_t \in R$.

There are different RL approaches in the specialized literature. Value-based solution methods [17] address these problems by learning value functions, i.e., the expected return for each state following a given policy. The value for a state can be estimated as the average of the return obtained over many episodes passing through that state. Temporal difference methods [18] such as SARSA and Q-learning [19] calculate this average incrementally and without waiting until the end of an episode: since the state value is defined recursively, they update one of their estimates at each time step from the immediate reward and the current estimate of the next estate. This generic update rule is defined as $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t))]$, where $V(S_t)$ is the current estimate of the value of the state $S_t \in S$ and α is a step size parameter.

When the state space is large or continuous it is not possible to estimate the value for all states. In these cases it is necessary to

use approximate methods that try to generalize the learned values by function approximation. Deep RL [20,21] involves approximate solution methods in which artificial neural networks (ANNs) are used as function approximators. In particular, a Deep Q-Network [22] is an approximate variant of Q-learning that uses an ANN to estimate the action-value function q_π. It modifies the update rule as a semi-gradient method. This algorithm incorporates two key aspects to solve the limitations associated to approximate methods:

- The algorithm uses a replay memory with past experiences, from which it extracts a random batch of training samples at each step. This reduces the correlation between consecutive samples and it smooths out abrupt changes in the data distribution.
- Second, it reduces the correlation between the estimates and the target values by using two difference ANNs: a behavior ANN which represents the current estimation of the action-value function and determines the action to be taken at each time step, and a target ANN which represents the target values (i.e., the value estimation of the next state in the update rule). These ANNs have the same architecture, but while the weights of the behavior ANN are updated continuously at each time step, the target ANN is only updated every few time steps as a copy of the behavior ANN.

With a tuple $e = (s, a, r, s')$ being a training sample, θ being the weights of the behavior ANN, θ^- being the weights of the target ANN, and $\hat{q}(s, a; \theta)$ being the estimate of the value of the pair (s, a) under the current policy from an ANN with the weights θ, the update rule for a sample is defined as in Eq. (1):

$$\theta \leftarrow \theta + \alpha\,[y - \hat{q}(s, a, \theta)]\,\nabla\hat{q}(s, a, \theta), \qquad y = r + \gamma \max_{a'} \hat{q}(s', a', \theta^-) \tag{1}$$

Deep Q-Network inherits a problem from the original Q-learning algorithm: it contains a positive bias in the value estimation of the action with the highest expected return. This bias is introduced by using the same estimates both to choose the action with the highest value and to determine its factual value. Double Q-learning [23] addresses this problem by using two separates estimators, one to determine the best action and the other to estimate its value. In this original version, the role of each estimator in each update is chosen randomly.

The DDQN algorithm [6] incorporates the idea of Double Q-learning into Deep Q-Network in a more direct way. As Deep Q-Network already uses two different estimators (the behavior and the target ANNs), it only needs a minimal change to decouple the best action from the estimation of its value: it only modifies the definition of the target value as in Eq. (2):

$$y = r + \gamma\hat{q}\left(s', \operatorname{argmax}_{a'} \hat{q}(s', a'; \theta); \theta^-\right) \tag{2}$$

In this version, the best action is still evaluated with the target ANN, but the change is to use the behavior ANN to choose the action to be evaluated as the best one. Despite being a minor modification, the results in [6] show that DDQN effectively reduces the positive bias in the estimates and achieves better policies than the standard Deep Q-Network.

There are diverse applications of deep RL in recent literature. Peng et al. [24] use RL techniques to learn the tone associated with each Chinese character. In this way, they can add the phonetic information of a character sequence to its visual and textual features, leading to better results in the Chinese sentiment analysis problem. On the other hand, Birman et al. [25] develop a method to use ensemble models more efficiently. Their proposal is to learn a policy through deep RL that dynamically determines which subset of the ensemble models offers the best trade-off between accuracy and computational cost for each sample of a binary classification problem.

2.3. Learning in ABMs

We can find previous works with different applications of machine learning in ABMs. van der Hoog [3] points out the possibility of using ANNs to create surrogate models that reduce the computational cost of the simulations, approximating the behavior of the ABM. The author proposes to reduce complexity at two levels: at local level, using different ANNs to emulate the behavior of each agent; or at global level, training a single ANN that directly estimates the ABM output from the initial values of the simulation.

The study also makes an additional proposal similar to what is done in this work, although without making a specific application: design agents that, instead of using predefined action rules, learn to make decisions through a battery of training simulations with the aim of maximizing a long-term reward. Thus, agents would have a dynamic behavior depending on the current state of the model. In addition, these agents would allow the ABM to be used to know the best possible action given a specific state of the environment, either in decision support systems or in game-theoretic models of social dilemmas and game theory [26].

Jäger [27] also addresses the idea of designing agents without predefined rules, but with a different approach. Instead of learning an optimal behavior, the idea is to create realistic agents, assuming that a real agent will follow a greedy strategy with immediate benefits. In this way, there is no need to use RL, considering instead a standard ANN to address the decision-making as a classification problem.

Previous works show specific applications of the idea of training agents in ABMs with RL in order to learn how to maximize a long-term reward. Collins et al. [28] model a insurgency scenario with three types of agents (soldiers, civilians, and rebels). They aim to train soldiers together to learn a behavior policy reducing the number of rebels and keeping as many oil assets as possible. In this ABM, the world is represented by a grid. The observation of a soldier are the squares around him, which may be free, occupied by another agent, or contain an oil asset. Therefore, the environment is discrete. Furthermore, the soldiers can only see three squares away, which makes the state space relatively small, so a tabular solution method such as SARSA can be used.

Hou et al. [29] use an ABM to simulate the process of collective cell migration, in which a leader cell moves towards a goal and guides a set of follower cells. Cells are trained with RL in order to learn to move in a similar way to reality, incorporating in the reward signal the constraints present in the real process. For instance, cells should not move too close to each other and proximity to target is favored. In this case the state space of the agents is their relative position and it is continuous, which makes necessary to use an approximate technique such as Deep Q-Network.

On the other hand, Sert et al. [30] use an ABM that simulates a social scenario with segregation dynamics in which two types of agents move around a grid world. Each agent observes at each moment their age and the 121 squares around them, which can have three possible values. This results in a state space too large to use tabular solution methods, so they also use Deep Q-Network. In this environment, each agent is trained with the aim to maximize a reward function that depends on factors such as aversion to agents of different type, interest in occupying the box of agents of different type to increase life time, and the fear of death. Authors perform experiments giving different weights to the different components of the reward function in order to analyze the resulting segregation dynamics in each scenario.

Other publications do not name ABMs explicitly but present multi-agent learning scenarios with some similarity to the studies mentioned above. For instance, Leibo et al. [31] and Schmid et al. [32] perform experiments on social dilemmas where different agents learn with independent Deep Q-Networks seeking individual rewards, but agents achieve the greatest benefits when they cooperate with each other.

In the literature we can also find applications of RL in marketing without the use of ABMs. Zhu et al. [33] explore the construction of MDPs in a marketing setting aiming to apply RL techniques to solve sequential targeting problems. On the other hand, Tian et al. [34] address the topic-aware influence maximization problem of viral marketing by defining a framework which combines graph embedding and RL. In more detail, it uses the DDQN algorithm to estimate the influence of each node in a social network under different situations.

As an alternative to all the latter works, we propose to train an agent that represents a brand in a marketing ABM to learn the best media planning strategy to maximize the brand awareness over time. Thus, the agent could be used as a decision support system if the ABM properly represents a real marketing scenario. To the best of our knowledge, this paper is the first attempt to use deep RL within an ABM calibrated with real data with the aim of assisting decision making in a real marketing scenario.

3. Description of the agent-based model

3.1. General structure

We use an ABM for replicating a market based on the works [5,35]. The model is run for a fixed number of w_{max} weeks of a marketing scenario involving three different types or levels of agents, each of them represented by a set:

- The set B represents the brands. They are high-level agents seeking to maximize its awareness $a^b \in [0,1]$ through advertising investments. The awareness value indicates the percentage of consumers who know the given brand in each week.
- The set M represents the touchpoints or media, intermediate agents that advertise brands according to the investments received.
- The set C represents a population of consumers. These agents can gain awareness of each brand in each week through social interactions and the impact of advertisements.

Brands and consumers influence each other: while the media planning decisions made by brands influence the topics of conversation of the consumers, the macro-level behavior of consumers will condition the awareness results obtained by brands and therefore their future decisions. Fig. 1 shows a summary of the dynamics of the presented ABM.

3.2. Population of consumers

Each consumer $c \in C$ has a binary variable $a_c^b \in \{0,1\}$ for each brand $b \in B$ that is updated in each week of the simulation. Specifically, $a_c^b(w) = 1$ indicates that consumer c knows brand b in week w, taking zero value otherwise. We initialize these values with a parameter $a^b(0) \in [0,1]$ which represents the fraction of consumers who are aware of brand b before the start of the simulation, satisfying that $a^b(0) = \frac{1}{|C|}\sum_{c=1}^{|C|} a_c^b(0)$.

Awareness has a dynamic behavior, so that each week any consumer can gain or lose its awareness of any brand. Thus, a consumer may discover a brand due to advertising or conversations with neighbors, but it may also lose its awareness values if these are not reinforced. This forgetting process is controlled by an awareness decay parameter $d \in [0,1]$ which indicates the probability that a consumer deactivates its awareness value for a brand in each week.

Consumers can activate its awareness values through two processes: the interaction with its neighbors and the impact of advertising. The first case is modeled as a diffusion process using an artificial social network that represents the conversations among the population of consumers, thus establishing the neighborhood of each consumer. We consider two alternatives to build this artificial social network. The first one is a random network, in which each new node is linked to the rest with an uniform probability [36]. The second alternative is to model a scale-free network with the Barabasi–Albert preferential attachment algorithm [37]. The degree distribution of this type of network follows a power law, where a majority of nodes have few connections and a few nodes (hubs) with a high degree serve as a bridge to traverse the network quickly. This second topology is present in many real networks, including social networks [10].

Once the network is defined, the diffusion process works as follows: in each week w every consumer c can start a conversation with its neighborhood with probability $p_c^b(w) \in [0,1]$ and talks about a brand b of which it is aware. This interaction is performed with all its neighbors simultaneously, each of them obtaining awareness of the brand involved according to a probability $\alpha^{WOM} \in [0,1]$ which represents the word-of-mouth (WOM) awareness impact.

3.3. Media investment through touchpoints

The second way to active awareness values of the consumers is through advertising, which depends on the decisions of the brands and the properties of the touchpoints. Each touchpoint is defined from a set of parameters that indicate aspects such as its reach and impact. In this way, we can model touchpoints with different nature, emulating that, for instance, a TV advertisement can only reach the portion of consumers who watch TV. On the one hand, the parameter that defines the reach of a touchpoint $m \in M$ is $r_m \in [0,1]$, which corresponds to the maximum percentage of the consumer population that the touchpoint m can impact in a week. On the other hand, the influence or impact is modeled with the parameter $\alpha_m \in [0,1]$, which indicates the probability that a consumer impacted by the touchpoint m at a given time gains awareness of the advertised brand.

In turn, advertising impacts can cause a viral buzz effect that increases the talking probability of the reached consumers. This effect depends on the buzz increment parameter τ_m associated with the touchpoint m who performed the advertisement. From the initial talking probability $p_c^b(0)$ and τ_m, we calculate the increase in the talking probability caused by $m \in M$ for $c \in C$ and $b \in B$ in week w as in Eq. (3):

$$\sigma_{cm}^b(w) = \sum_{s=1}^{w} \left(p_c^b(s) - p_c^b(0) \cdot \tau_m\right) \tag{3}$$

However, this buzz effect can be reduced over time through a buzz decay parameter $d\tau_m$. Bringing together all the above, the complete process that regulates the talking probability in each week of the simulation is defined in Eq. (4).

$$p_c^b(w+1) = p_c^b(w) - \sigma_{cm}^b(w) \cdot d\tau_m + p_c^b(0) \cdot \tau_m \tag{4}$$

Finally, the number of impacts that each touchpoint makes in a week for a given brand depends on the investment made by that brand. Specifically, the investment is measured in Gross Rating Points (GRPs), with one unit of this measure representing the reach of 1% of the target consumer population. Each brand has a fixed budget $\phi^b(w)$ for each week w, and must decide how to distribute it among the different touchpoints. The variable that indicates the investment made by a brand b in week w in the touchpoint m is $\chi_m^b(w)$, satisfying that $\phi^b(w) = \sum_{m=1}^{|M|} \chi_m^b(w)$.

4. Learning of investment policies by brand agents

From the ABM previously described and considering a brand of interest $\tilde{b} \in B$ and a limited and predefined budget for a specific number of weeks $\phi^{\tilde{b}}$, we define the tackled problem as finding the best advertising investment strategy to be followed by brand $\tilde{b}$ to maximize its awareness $a^{\tilde{b}} \in [0,1]$ over time. The best policy is not only the one that increases the awareness as soon as possible, but also that keeping high values until the end of the simulation.

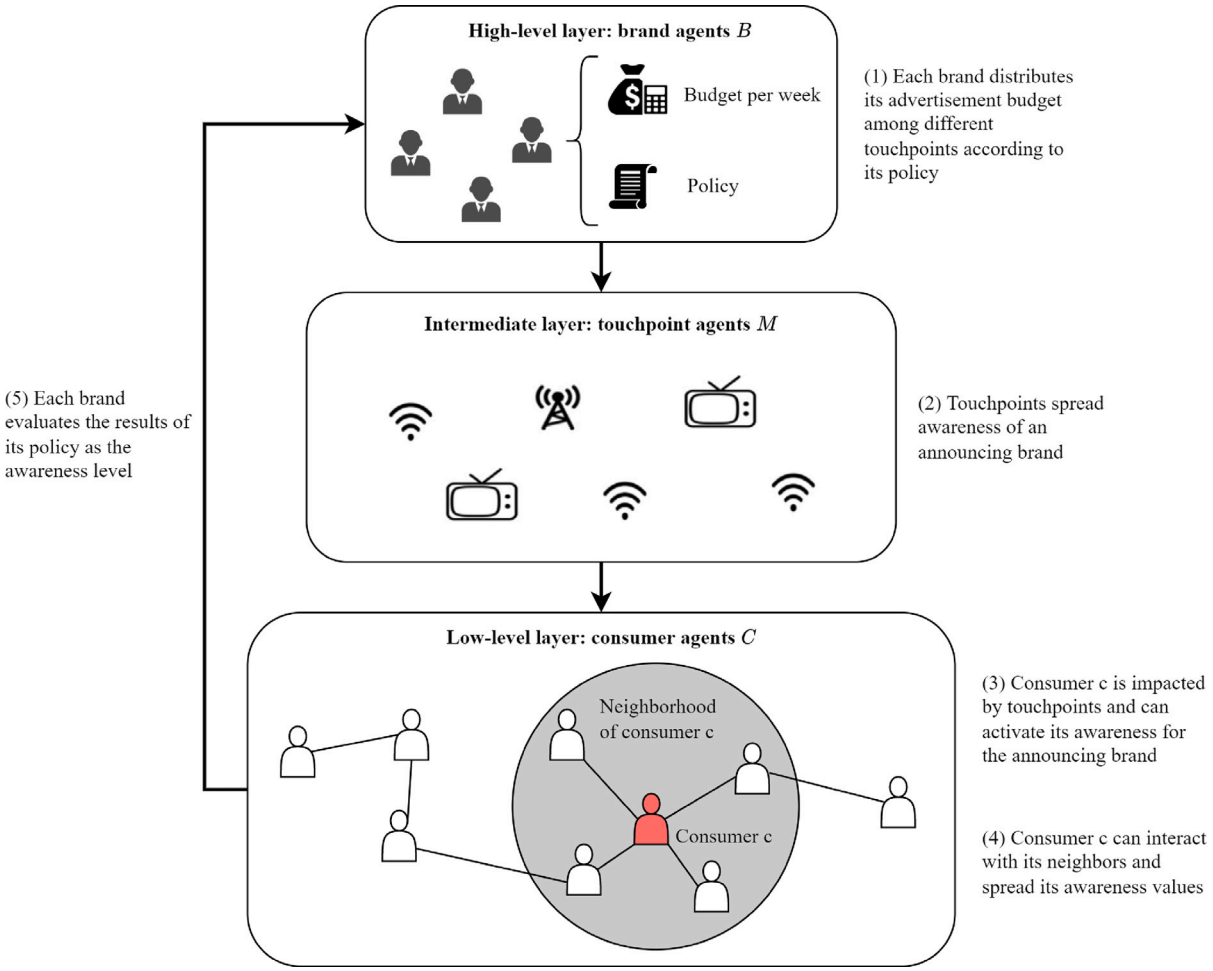


Fig. 1. Summary of the ABM operation. Each week, brands invest in different touchpoints and consumers can be influenced by advertising, gain awareness of the brand announced, and spread it to their neighbors. Brands evaluate their decisions depending on the resulting awareness level within the consumer population.

Traditional marketers usually define static investment policies, i.e, long-term marketing investment plans for one or two years. In fact, as we will detail in Section 5.1, we have data of an actual static strategy followed by brand $\tilde{b}$ which was defined in advance for the entire simulation time. This strategy will serve as a baseline to evaluate the alternative policy developed in this work. Our proposal is to define a dynamic investment policy that is able to adapt to the market evolution every few weeks (i.e., a time window). In this way, it would be possible to act in response to market changes that are difficult to anticipate in the long term.

However, even using shorter time lags between decisions, it is difficult to determine the optimal investment distribution to make at each time window, since this will condition the resulting market state when making the next decision. For this reason, we propose to address this problem with RL, which is suitable for problems where the effect of an action affects not only to the immediate reward, but also to the next situation and hence future rewards. Consequently, our idea is to train a RL agent representing brand $\tilde{b}$ through a battery of ABM simulations with the objective of learning a dynamic policy that maximizes its long-term awareness. The brand agent interacts with the environment every time window, and in each interaction it must decide the media planning for the next time window given a fixed budget.

It is worth noting that in reality some advertising media require long-term investment allocations. Therefore, we define two types of touchpoints: off-line touchpoints M_{off}, in which the investment is decided yearly; and on-line touchpoints M_{on}, which admit short-term investments. The brand agent only considers on-line touchpoints when defining the dynamic advertising investment strategy.

The first step to address this problem with RL is defining the underlying MDP, in particular the state space, the action space, and the reward function. Fig. 2 shows a summary of the interaction process between the learning brand agent and the model by including all the elements of the MDP. It is important to distinguish between the sequence of steps of the MDP ($t = 0, 1, 2, \dots, T$) and the simulation steps of the ABM ($w = 1, 2, 3, \dots, w_{max}$). While the steps of the MDP indicate the moments of the interaction of the brand agent, the steps of the ABM denote the current week of the simulation. We define W as the size of the time window (the number of weeks it comprises), and Z_t as the time window over which action A_t is applied. The corresponding simulation week to $Z_t(i)$ is calculated as $w = t \cdot W + i$, where $t \in \{0, \dots, T-1\}$ and $i \in \{1, \dots, W\}$.

The state space S defines all the ABM variables that can change through the simulation. However, to look for an appropriate trade-off between completeness and convergence, we eliminate variables that do not provide information of interest for the problem to be solved. The chosen state space is formed by the following $3 + |M_{\text{on}}|$ real values, where $|M_{\text{on}}|$ is the number of on-line touchpoints in the market considered:

- β_t: Total budget available during the current time window Z_t. This amount can be important to determine how many touchpoints to invest in. While a low budget makes it more appropriate to focus investment on the highest impact touchpoints, a higher budget allows the learning agent to invest in more touchpoints to ensure reaching a wider range of consumers.

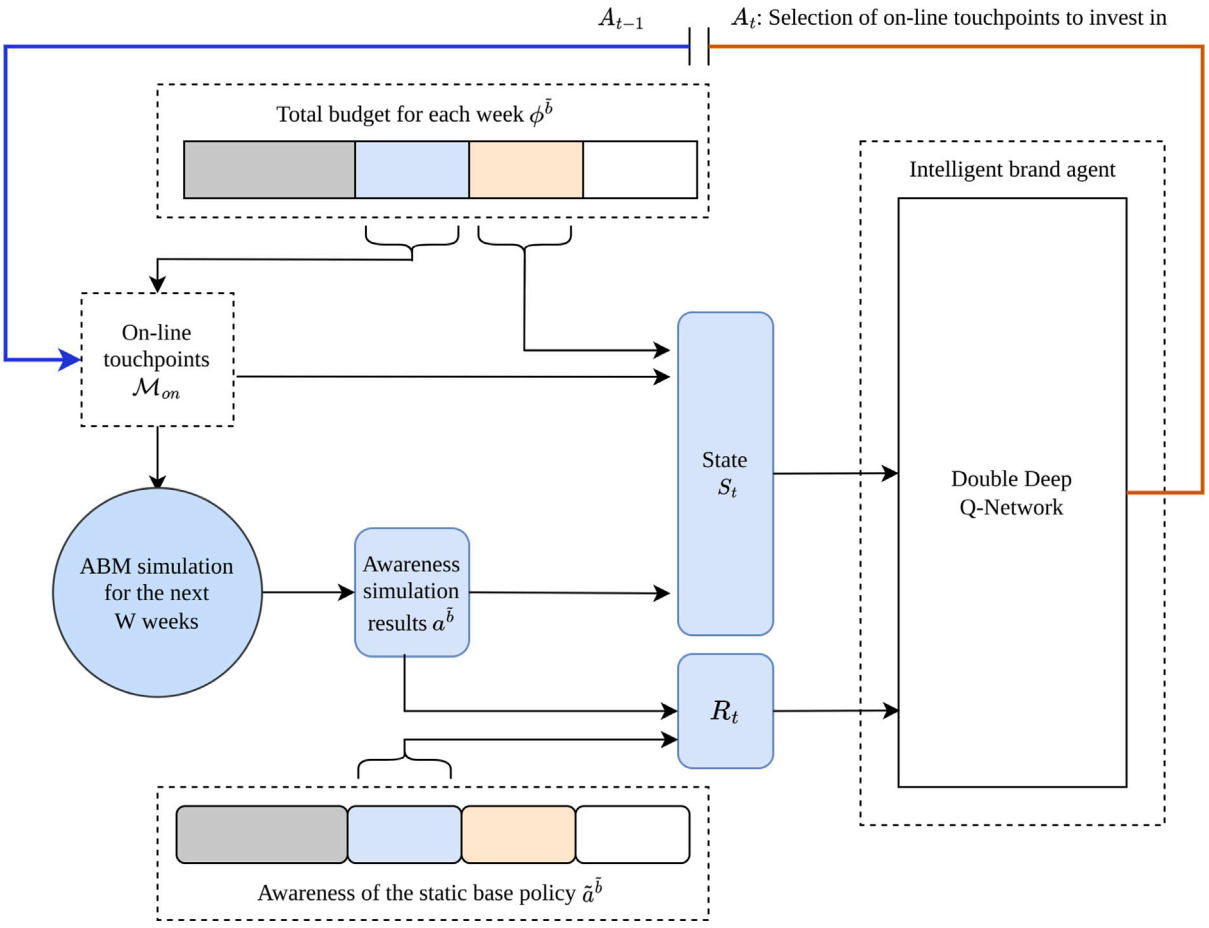


Fig. 2. Summary of the RL brand agent operation. At each time step t, the brand agent receives the reward R_t of its previous action A_{t-1} and the resulting new state of the environment S_t. Gray color is associated with the simulation weeks prior to time window Z_{t-1}. Blue color is associated with the application of the previous action A_{t-1} over the weeks corresponding to time window Z_{t-1}, whose results are the state S_t and the reward R_t. Orange color is associated with action A_t, which will be applied over the next time window Z_t. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\beta_t = \sum_{w=Z_t(1)}^{Z_t(W)} \phi^{\tilde{b}}(w) \quad \forall t \in \{0, \dots, T-1\}$$

- N_t: Difference between the current awareness and the initial awareness of the simulation. This variable indicates the current state of the brand awareness and it is normalized with respect to its initial awareness, making it independent of the initial simulation conditions. Thus, if the current awareness is 0.6 but the initial awareness was 0.7, this value will be negative, indicating a declining awareness state. It may be appropriate to perform more or less aggressive actions depending on the current situation of the brand.

$$N_t = a^{\tilde{b}} Z_{t-1}(W) - a^{\tilde{b}}(0); \quad \forall t \in \{1, \dots, T-1\}; \quad N_0 = 0$$

- D_t: Difference between the current awareness and the initial awareness of the previous time window. This difference allows the agent to know the awareness trend (either increasing or decreasing). The current awareness dynamics could influence the choice of the best budget distribution.

$$D_t = a^{\tilde{b}} Z_{t-1}(W) - a^{\tilde{b}} Z_{t-2}(W); \quad \forall t \in \{2, \dots, T-1\};$$
$$D_0 = 0; \quad D_1 = N_1$$

- I_t^m: Total amount of investment made until the step t in each on-line touchpoint m. Investments are normalized with respect to the total budget of the whole simulation period ($\sum_{w=1}^{w_{max}} \phi^{\tilde{b}}(w)$), thus taking values in range $[0, 1]$. It is useful to know how much has been invested in each touchpoint to decide whether to invest in the touchpoints that have been neglected to invest so far.

$$I_t^m = \sum_{w=1}^{Z_{t-1}(W)} \chi_m^{\tilde{b}}(w); \quad \forall m \in M_{\text{on}} \quad \forall t \in \{1, \dots, T-1\}; \quad I_0^m = 0$$

The action space A could be the proportion of the available budget to be assigned to each touchpoint. However, this would result in a continuous action space, difficult to be handled by the RL algorithm. Instead, we look for a balance between the size of the action space and the accuracy of the designed strategy by considering the following action space: at each time window, the agent has to choose in which touchpoints to invest (a binary coding), distributing the available budget equally among the chosen ones during the weeks corresponding to the current time window. Thus, the number of possible actions is $2^{|M_{\text{on}}|} - 1$. The latter value is the result of all possible combinations of activating or deactivating the investment for each on-line touchpoint, except the combination that does not perform any inversion.

The most straightforward reward function is the awareness obtained over the last time window. However, we can take advantage of the knowledge given by the previously mentioned static policy based on the actual advertising investment applied by the brand, so that the objective of the agent is to outperform that static baseline policy as much as possible. In this way, the reward will measure the difference between the awareness obtained and that corresponding to the actual

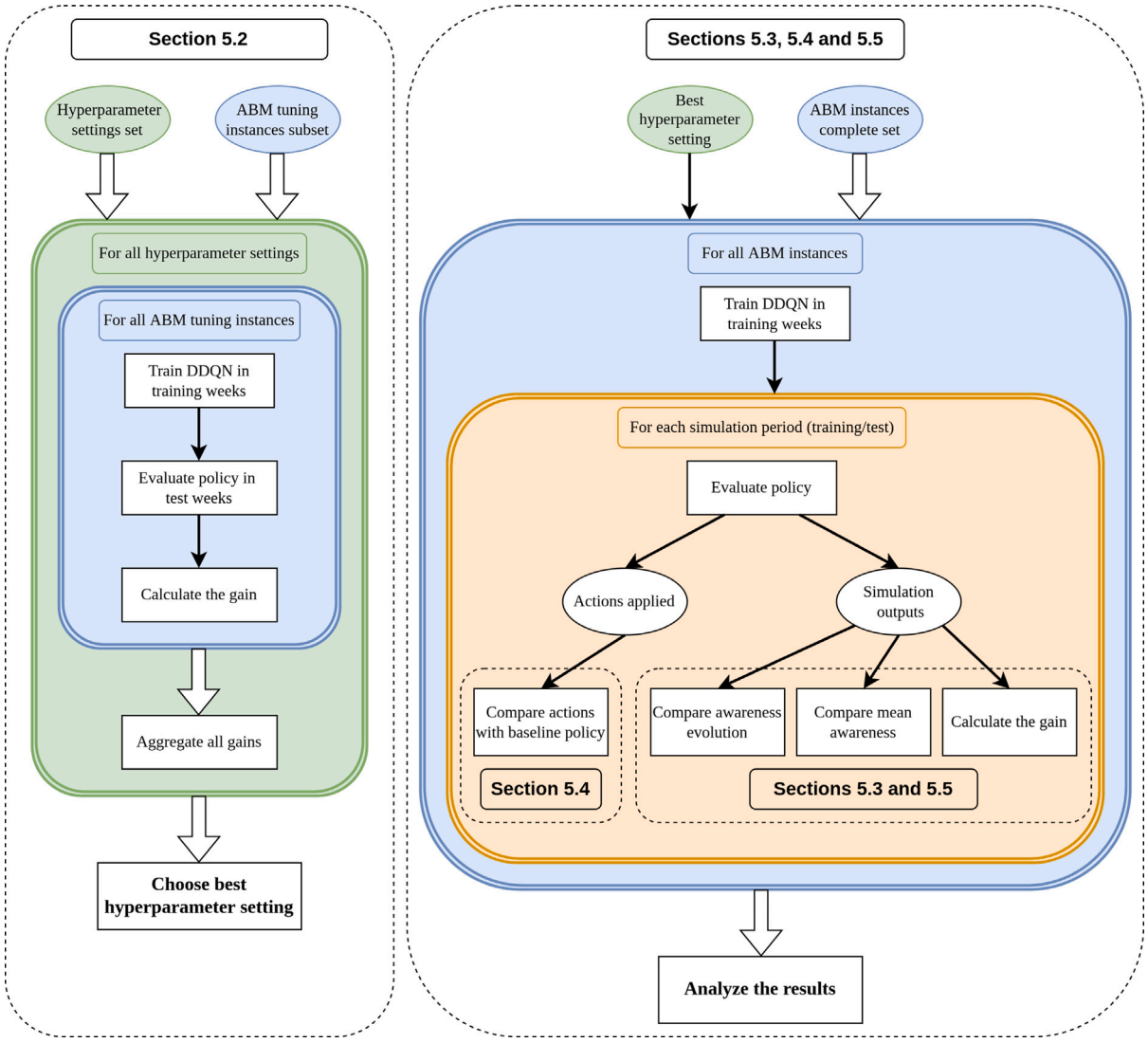


Fig. 3. Flowchart diagram of the experimentation. First, we perform the hyperparameter tuning process illustrated on the left considering a set of hyperparameter settings and some of the ABM instances. Then, as illustrated on the right, we use the best hyperparameter setting found to train a DDQN brand agent in each ABM instance and we evaluate the learned dynamic policy by comparing its results with those given by the static baseline policy. Finally, we also compare the awareness results for the baseline instance with those obtained by an econometric model-based policy.

investment policy made by the brand. Thus, rewards take negative values when the awareness results obtained are lower than those of the reference policy. With $Z_t(1)$ being the first week of the current time window, $Z_t(W)$ being its last week, $\tilde{a}^{\bar{b}}(w)$ being the awareness value obtained by the static baseline policy in week w, and $a^{\bar{b}}(w)$ being the awareness obtained by the agent, the immediate reward is performed according to Eq. (5).

$$R_{t+1} = \frac{1}{W} \sum_{w=Z_t(1)}^{Z_t(W)} \left(a^{\bar{b}}(w) - \tilde{a}^{\bar{b}}(w) \right) \tag{5}$$

An episode ends when the last week of the simulation is reached, and the return of the episode is the discounted sum of the rewards obtained in each time window.

The defined problem has a continuous state space and the agent does not have a model of the environment, i.e., the agent does not know the dynamics of the environment. Therefore, we have to use an approximated and model-free solution method. As detailed in Section 2.2, Deep Q-Network [22] is a deep RL algorithm that solves the stability problems of approximated methods by means of a replay memory with past experiences and using two different estimators. Moreover, in Section 2.3 we showed the popularity of this method, which is the most used in recent literature for problems with continuous spaces such as ours. For all these reasons we decide to use this learning algorithm. Concretely, we make use of the DDQN [6] variant explained in Section 2.2. The DDQN variant achieves better results than the original version through a small adjustment in the value estimation of the best action.

5. Experimentation

The experimentation comprises different runs with the DDQN algorithm on a set of seven instances of the ABM system. In addition, each of these instances has two different simulation periods: one for training and one for testing. In this way, we only train the learning agent over the training weeks, and we evaluate the results of its learned dynamic policy through separate simulations in the test weeks. Furthermore, we compare the results obtained by the agent in each instance with those given by a real static strategy previously used by the brand of interest.

The rest of the section is structured as follows. First, in Section 5.1, we show the experimental setup and describe the seven instances of the model. Second, in Section 5.2 we detail the values chosen for the DDQN hyperparameters, performing a tuning process for some of them. Then, we use the specified configuration to train a brand agent for each of the seven instances of the model. We show the awareness

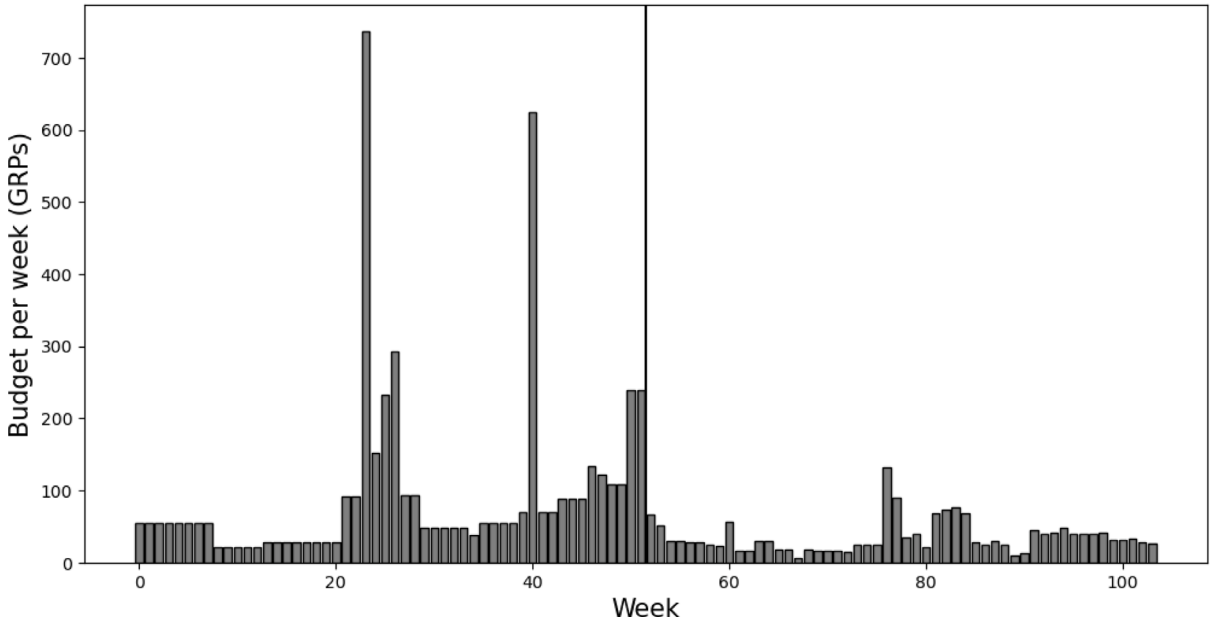


Fig. 4. Available budget per week to allocate among on-line touchpoints in the Bank scenario. The vertical black line separates the training and test weeks.

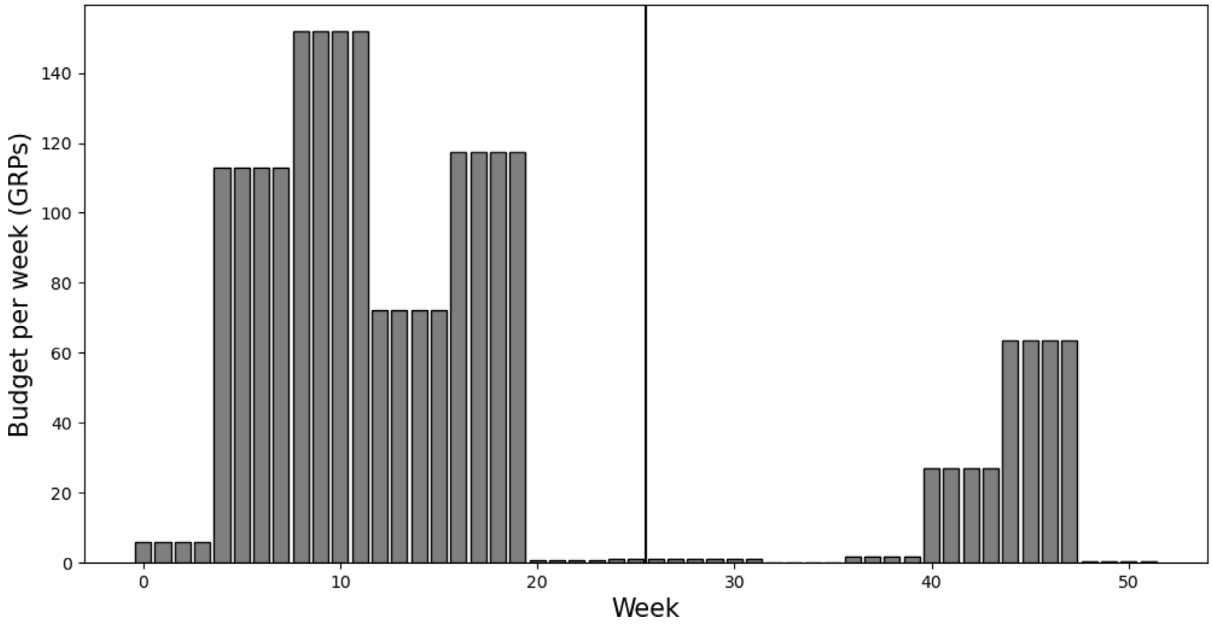


Fig. 5. Available budget per week to allocate among on-line touchpoints in the Service scenario. The vertical black line separates the training and test weeks.

results of the trained agents and the comparison with those given by the static baseline strategy in Section 5.3. Next, we analyze the investment policies made by these agents in Section 5.4. We show the flowchart diagram of this experimentation in Fig. 3. Finally, in Section 5.5, we present a static strategy based on an econometric model and compare it with the learned DDQN policy.

5.1. Experimental setup and instances of the model

We have used a cluster with 4 Intel(R) Core(TM) i7-4930K CPU @ 3.40 GHz (6 cores) and 64 GB RAM to perform every experiment. Such resources are necessary due to the high computational cost required for each training activity, where each episode involves a complete simulation of the ABM in the specified period. The hyperparameter tuning described in Section 5.2 is the most computationally costly part, where 162 training processes are performed with between 7,692 and 14,286 episodes each of them. The top layer of the source code, which implements the RL mechanisms of the proposal, is publicly available at https://github.com/vvarper/drl4mediaplanning to facilitate the application of our method in other domains.

In order to test the robustness of the method, we train the brand agent in two different variants of the marketing ABM. Both variants are built on data from real marketing scenarios, although they are anonymized. The most relevant features of each variant are as follows:

- The Bank scenario (referred as B in the tables and instances employed) simulates a bank marketing environment for credit card sales. In this scenario, there are 40,000 consumers connected by a random network and a set of 9 brands which perform advertising investments on 9 different touchpoints, five of them

Table 1
Available budget in both scenarios for each time window and its percentage with respect to the total training and test budget. The total available budget in the Bank scenario is 5107.82 GRPs in the training period, while in test it is 1844.34 GRPs. The total available budget in the Service scenario is 1846.63 GRPs in the training period, while in test it is 374.58 GRPs.

Period	Bank scenario			Service scenario		
	Window	Budget (GRPs)	Percentage (%)	Window	Budget (GRPs)	Percentage (%)
Training	0	440.13	8.62	0	475.98	25.78
	1	192.63	3.77	1	896.34	48.54
	2	1.059.88	20.75	2	472.73	25.6
	3	1.009.11	19.76	3	1.58	0.09
	4	424.54	8.31			
	5	1.285.41	25.17			
	6	696.13	13.63			
Test	7	282.88	15.34	4	4.77	1.27
	8	189.22	10.26	5	60.4	16.12
	9	155.9	8.45	6	308.58	82.38
	10	536.46	29.09	7	0.84	0.22
	11	242.02	13.12			
	12	319.04	17.3			
	13	118.81	6.44			

on-line (touchpoints 0, 2, 6, 7, and 8). This model considers 104 weeks of simulation.

- The Service scenario (referred as S in the tables and instances employed) corresponds to a service company case. There are 1,000 consumers connected through a scale-free network, 8 brands, 7 touchpoints, and 52 weeks of simulation. Five of the touchpoints are on-line, specifically numbers 0, 2, 4, 5, and 6.

The simulation period of each of these scenarios has been splitted into two subsets to validate the learning process. All the learning process described in the following sections is performed on the training weeks, which correspond to the first half of the simulation period: 52 weeks for the Bank scenario and 26 weeks for the Service scenario. We use the second half of the simulation period to evaluate the trained agent, testing whether it is able to generalize the learned policy to unknown states.

The available budget for the brand of interest to allocate in on-line touchpoints accounts more than 90% of its total marketing budget in both scenarios. Therefore, although there is a small amount of fixed investment dedicated to off-line touchpoints, the distribution of the on-line budget is the most decisive decision in the final awareness result. Figs. 4 and 5 show the on-line budget per week in the Bank and Service scenarios, respectively.

As we will detail in the following section, the chosen version of the learning agent uses time windows of 8 weeks (W). For this reason, we show the available budget for both scenarios considering time windows of this size in Table 1. This table contains both the available amount at each time window and the percentage that this amount represents with respect to the total budget, either in training or test weeks.

We have a static baseline policy for both scenarios previously used by the real brand. Unlike the learning agent, the investments of this static policy are not defined by time windows, but for the entire simulation period. In addition, the distribution at each week does not have to be uniform across all the chosen touchpoints. However, in order to make easier to compare these baseline policies with the dynamic policies learned by the brand agent in the following sections, we show in Figs. 6 and 7 the investment distribution of the baseline policies in time windows of 8 weeks. The same plots show the investment for both training and test weeks, which are divided by the vertical black line in the center. In more detail, these plots show the percentage of the available budget assigned to each touchpoint for each time window. Thus, for example, in Fig. 6 we can see that the whole available budget for windows 0 and 1 is assigned to touchpoint 8, while in window 2 most of the budget is assigned to touchpoint 0, and the remaining amount is divided between touchpoints 2 and 8.

So far we have described the details of these two real marketing scenarios and the advertising investment policies originally applied in them by the brand of interest. However, to test the robustness of the learning method on problems with awareness dynamics of different complexity, we have generated a total of seven instances of the model. These instances are defined as variants of the above real scenarios with modifications in the parameters of awareness deactivation d and each on-line touchpoint impact α_m. We show the specific parameter setting of each of these instances in Table 2.

Table 2
Awareness deactivation and touchpoint impact parameters in each instance.

Instance	d	α_0	α_2	α_6	α_7	α_8
B-original	0	0.0045	0.0045	0.7	0.6	0.0095
B-002bal	0.02	0.4	0.35	0.5	0.45	0.4
B-002un	0.02	0.0045	0.0045	0.9	0.85	0.75
B-01bal	0.1	0.4	0.35	0.5	0.45	0.4
B-01un	0.1	0.0045	0.0045	0.9	0.85	0.75
Instance	d	α_0	α_2	α_4	α_5	α_6
S-001a	0.01	0.01	0.4	0.25	0.65	0.01
S-008b	0.08	0.01	0.02	0.7	0.25	0.54

There are 5 instances of the Bank scenario. The first one has the original calibrated parameters that correspond to the real marketing scenario, where agents do not forget their brand awareness values. The other four instances make the problem more difficult by increasing the deactivation value to 0.02 and 0.1 (a low and a high value, respectively) and considering two configurations for the touchpoint impact values: one unbalanced (un) in favor of touchpoints 6, 7, and 8; and another more balanced (bal) where the impact of all touchpoints is relatively similar. It will be of interest to test whether the brand agent tends to spread the investment more widely or to concentrate it in a few touchpoints according to these configurations.

With respect to the Service scenario, the original calibrated configuration is omitted because all touchpoints have excessively low impact values, so that the relevance of the investment distribution decreases. Instead, we use two synthetic variants with deactivation values 0.01 and 0.08 (low and high, respectively), and with unbalanced touchpoint impact values.

The original investment (i.e., the static baseline policy) is the same for instances of the same scenario, since it is the one performed in the corresponding real scenario (Figs. 6 and 7). However, the reference baseline awareness is different for each instance, as it corresponds to the simulation output of that instance using the associated static baseline policy. The aim of the brand agent is to learn how to redistribute the available investment budget in order to obtain results that improve those of the static baseline policy as much as possible.

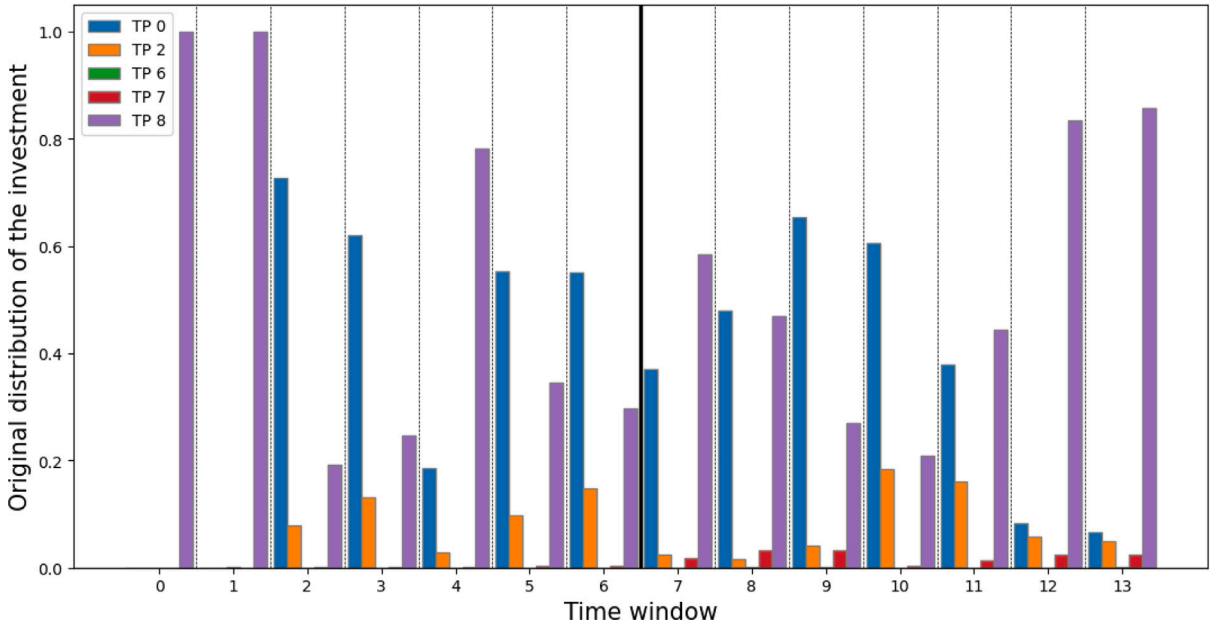


Fig. 6. Investment distribution among the touchpoints (TP) for each time window of the original media planning (static baseline policy) in the Bank scenario. The vertical black line separates the training and test windows.

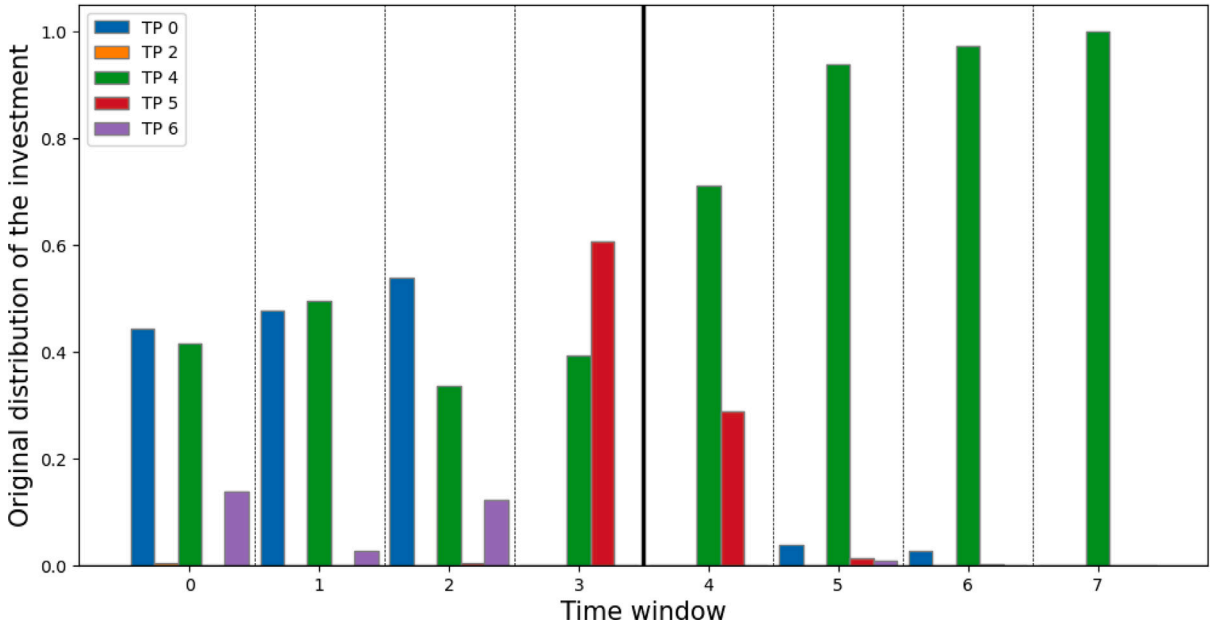


Fig. 7. Investment distribution among the touchpoints (TP) for each time window of the original media planning (static baseline policy) in the Service scenario. The vertical black line separates the training and test windows.

5.2. Hyperparameter tuning

The DDQN algorithm has a large number of hyperparameters, which makes it difficult to determine the best configuration. An exhaustive search would involve a large number of learning processes on each instance, which is unfeasible due to the high computational cost of training with a sufficient number of episodes to achieve effective learning.

For this reason, we set many of the hyperparameters to reasonable values. First, we define those related to the ANN design. Since the state space does not incorporate matrix information, we use a multi-layer perceptron in which the number of input and output units is determined by the dimension of the state space $3 + |T_{\text{on}}|$ and the number of possible actions $2^{|T_{\text{on}}|} - 1$. On the other hand, the ANN has 3 hidden layers (hierarchy in knowledge extraction) with 64 units each and the activation function ReLU. In this way, the number of neurons in the hidden layers is higher than in the output layer, which in our case contains $2^5 - 1 = 31$ neurons because we have 5 on-line touchpoints in all instances. We choose the Adam optimization algorithm, whose good results are shown in [38].

We set the number of steps to 100,000, since we found in a preliminary training set that this value provides a good trade-off between the quality of learning (convergence) and the time required. We fix most of the specific hyperparameters of Deep Q-Network using as a reference the values indicated by [22], but taking into the account that our training involves 125 times less steps. Considering all the latter, we set the update frequency of the target network to 800 steps and use a

Table 3
Subset of hyperparameters to estimate together with its description and the candidate values considered.

Hyperparameter	Description	Candidate values
Time window size W	Frequency in weeks in which the agent makes a decision.	4, 6, and 8 weeks
Batch size	Number of past samples used on each SGD update.	16, 32, and 64 samples
Replay memory size	Maximum memory size of recent past samples. The batch of each iteration is sampled from this memory.	10,000, 20,000, and 30,000 samples
Discount factor γ	Parameter of Q-learning updates that indicates the importance of future rewards.	0.9, 0.95, and 0.99

ϵ-greedy behavior policy whose exploration probability ϵ is uniformly reduced from 1 to 0.1 during the first 20,000 training steps.

The remaining hyperparameters are the replay memory size, the batch size, and the discount factor γ. We perform a tuning of these hyperparameters together with the time window size W with the same approach as in [22], where it is performed a search for the best hyperparameter setting on a reduced number of variants of the environment, which is used then to train a brand agent on the rest of variants. In our case, we use two instances for the tuning process: B-002un and B-01bal. Although they belong to the same scenario, they have different touchpoint impact and deactivation parameters, and therefore different awareness dynamics. Table 3 shows the description of each of these hyperparameters together with the candidate values considered.

The tentative values for the time window size were chosen considering that in a real marketing scenario it is unusual to make advertising investments for time periods shorter than one month, even with online touchpoints. This way, we consider a time window between one or two months. The values of the remaining hyperparameters were chosen taking typical values from the RL literature and using [22] as main reference, as it was done for the fixed hyperparameters.

This search comprises $3^4 = 81$ settings. Since we use two instances of the ABM, we perform a total of 162 training runs, each of them requiring between 20 and 40 h of computation. This variation depends on the size of the time window, since the larger the window, the fewer the number of steps per episode, and hence the greater the total number of episodes (full model simulations) needed to achieve the 100,000 training steps.

Once a brand agent finishes its learning process, we evaluate its performance by calculating the expected gain as the average gain of 30 different simulations in test weeks (i.e., a Monte Carlo approach, usually followed with ABMs). With w_{max} being the number of weeks of an ABM instance simulation (either training or test weeks), $a^{\tilde{b}}(w)$ being the awareness obtained by the RL agent in week w, and $\tilde{a}^{\tilde{b}}(w)$ being the awareness corresponding to the static baseline policy, we define the gain of the trained brand agent as in Eq. (6).

$$gain = \frac{1}{w_{max}} \sum_{w=1}^{w_{max}} \left(a^{\tilde{b}}(w) - \tilde{a}^{\tilde{b}}(w) \right) \tag{6}$$

Although they have a similar definition, this gain should not be confused with the return G_t of the RL theory, which in our problem is defined from the immediate reward function (Eq. (5)) described in Section 4. The return (i.e., the discounted sum of the rewards obtained in each time window) is used to guide the RL agent during its training process. However, we evaluate the quality of the learned policy with the gain because it offers greater interpretability when comparing its awareness results with the static baseline policy.

It should be noted that it is not possible to directly compare the results between different instances, as the scale of their gains varies. For this reason, we apply a standard normalization to zero mean and unit variance to the 81 test results for each instance. We use this normalized result as the score of each hyperparameter setting in the corresponding instance and we calculate the final score of each setting as the mean for both instances. Table A.5 in Appendix A shows the average test gain in each instance and the final score for each hyperparameter setting, with entries ordered from best to worst according to the final score.

The best setting differs for each instance. In the instance B-002un there is a draw between three settings which achieve the highest average test gain (0.1504) and score (0.912). However, all these three settings have poor results in the B-01bal instance, with even the highest score of them being negative in this instance (−0.6373).

On the other hand, the best setting in B-01bal instance achieves an average test gain of 0.066 and a score of 1.3012, being also the setting with the highest final score (0.9576). It might be thought that the choice of this setting tips the balance too much in favor of instance B-01bal. However, the average test gain of this setting in B-002un is 0.1486, which is not much lower than the higher value found (0.1504) considering than the lowest average gain in this instance is 0.1059. Therefore, we consider the setting with the highest final score to be good enough in both scenarios, and hence we set the time window size to 8 weeks, the batch size to 64 samples, the replay memory size to 20,000 samples, and the discount factor γ to 0.95 for the training runs in the remaining experiments.

5.3. Awareness results of the brand agent

The second part of the experimentation involves training a brand agent in each of the seven instances of the model with the previously indicated hyperparameter setting and evaluating its results against the static baseline policy. The training time in each Bank instance is around the longest time of the previous experimentation (40 h), as we use the larger time window size (8 weeks). Training in Service instances are much faster, requiring 1 h each of them. There are two reasons for this dramatic difference: first, the simulation period for training in these instances is shorter, 26 weeks instead of 52. Second, the cost of each full simulation is much lower, since the Service scenario contains 40 times fewer consumers than the Bank scenario.

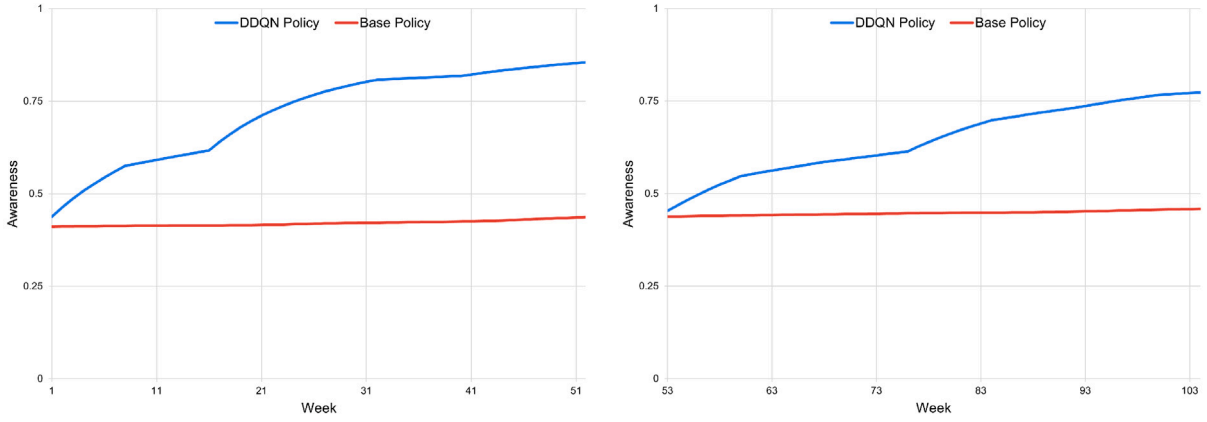
Figs. B.13 and B.14 in Appendix B show the return evolution throughout the training process for instances B-original and B-002bal. Each full simulation of the ABM uses a different pseudo-random seed from a set of 30, so we perform the return mean of each 30 consecutive episodes for the visualization. In all cases the learning reaches convergence around step 20,000, i.e., the time when the *ϵ-greedy* behavior policy reaches the minimum ϵ of 0.1.

Table 4 shows a comparison between the results obtained by each trained agent and those obtained by the static baseline policy. Specifically, we show the mean awareness obtained by each policy over 30 simulations in the training and test weeks separately. We also show in this table the corresponding gain of the learning agent (Eq. (6)), a measure that directly indicates the amount of improvement over the baseline policy. In all instances the gain is greater in training weeks, although the gain in the test period is still positive: the brand agent has learned a policy in all scenarios that, to a higher or lower extent, outperforms the original baseline policy even in previously unseen situations.

Next we show a graphical comparison between the awareness results obtained by both policies, showing the awareness evolution

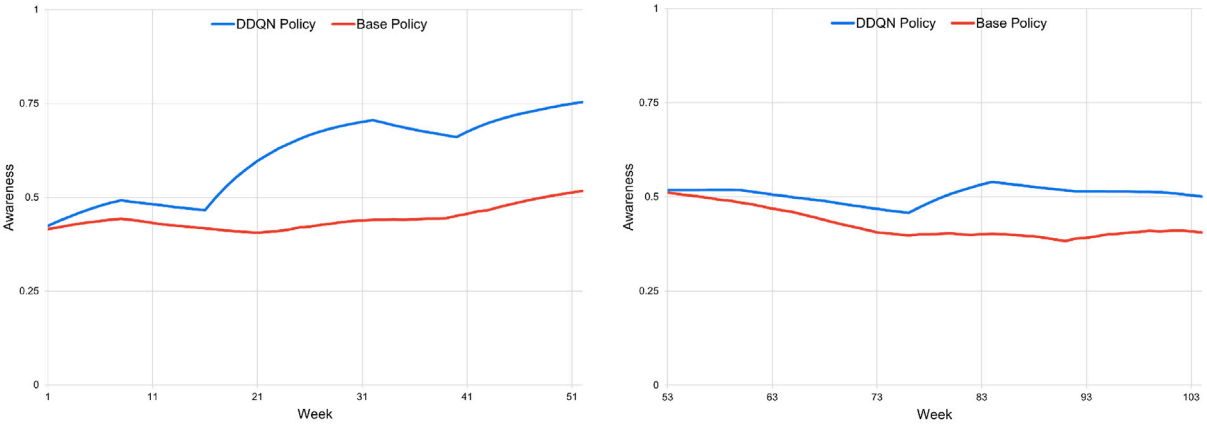
Table 4
Mean awareness obtained by the trained DDQN agent and the static baseline policy in training and test weeks in all instances. The best training and test values for each instance are highlighted in bold. The last two columns contain the resulting gain obtained by the brand agent with respect to the baseline policy.

Instance	Training mean awareness baseline policy	Training mean awareness DDQN	Test mean awareness baseline policy	Test mean awareness DDQN	Training gain	Test gain
B-original	0.420	**0.718**	0.447	**0.645**	0.2982	0.1978
B-002bal	0.442	**0.610**	0.427	**0.506**	0.1674	0.0798
B-002un	0.472	**0.675**	0.434	**0.583**	0.2027	0.1486
B-01bal	0.202	**0.359**	0.119	**0.185**	0.1576	0.066
B-01un	0.223	**0.431**	0.132	**0.257**	0.208	0.1253
S-001a	0.124	**0.231**	0.168	**0.182**	0.214	0.0289
S-008b	0.127	**0.184**	0.068	**0.073**	0.1125	0.0102



(a) Training weeks (b) Test weeks

Fig. 8. Awareness output of simulation in B-original instance.



(a) Training weeks (b) Test weeks

Fig. 9. Awareness output of simulation in B-002bal instance.

throughout the simulation time. Figs. 8 and 9 show the evolution for instances B-original and B-002bal, both in training and test weeks. The results in the remaining instances can be found in Figs. C.15 to C.19 in Appendix C. In all cases the awareness obtained by the trained agent outperforms that of the static baseline policy. This is particularly relevant in the B-original instance, where the parameters of the model correspond to a real scenario. In fact, the highest improvement is found in this instance, achieving awareness values above 80% in the training weeks and above 70% in the test.

The results are not as good in the synthetic instances despite the improvement with respect to the original investment policy values. This may be due to the increase in the complexity of the scenario because of the incorporation of the awareness deactivation parameter. In fact, in the Bank instances, the worst results are found in the cases with the highest deactivation, 0.1. Still, in terms of gain, the touchpoint impact setting seems to be the most determining factor: the gain in the balanced instances (bal) is lower than in the unbalanced ones (un). This may be due to the fact that the margin for improvement redistributing the budget is smaller when all touchpoints have a similar impact.

If we compare the results in both scenarios, the Service scenario clearly shows the worst performance, specially in instance S-008b. The reason could be that this is a more complex environment where it is more difficult to improve the results by simply redistributing the given budget. Nevertheless, we recall that the policy learned by the agent still improves the results obtained by the baseline advertising investment policy even in this case.

Finally, it is worth noting that, as seen in Table 1 (as well as Figs. 4 and 5), the available budget in the training weeks is substantially higher than in the test weeks, which is determined by the policy that the brand of interest applied in the real scenario. This difference in the budget is particularly relevant in the Service scenario, where the budget in test is seven times lower than in training. This could

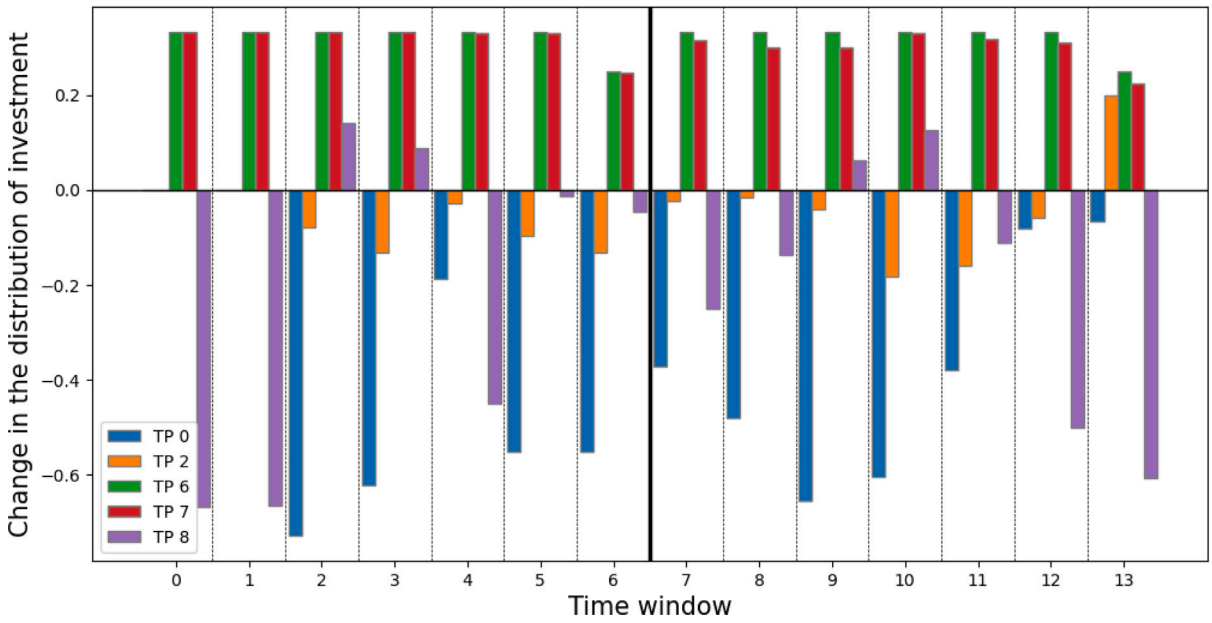


Fig. 10. Alteration of the investment distribution among the touchpoints (TP) by the brand agent in the B-01un instance. The vertical black line separates the training and test windows.

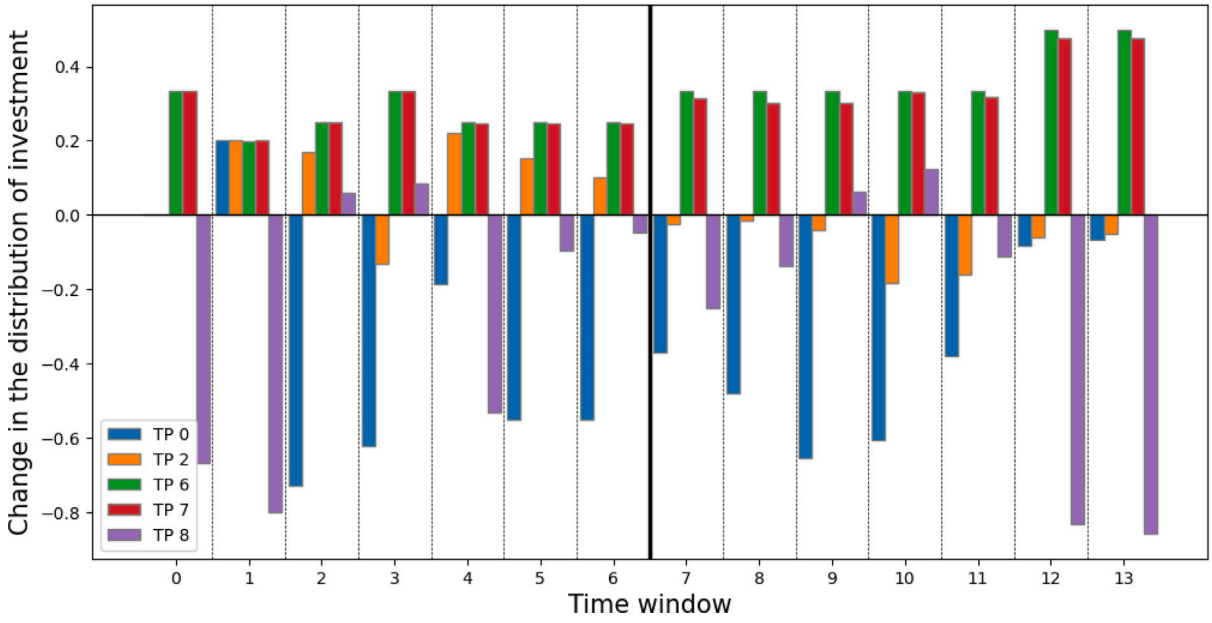


Fig. 11. Alteration of the investment distribution among the touchpoints (TP) by the brand agent in the B-002bal instance. The vertical black line separates the training and test windows.

explain the worse performance of the brand agent in test: the problem would not be so much the learned policy as the available budget, which prevents a significant improvement in awareness regardless of the budget distribution.

5.4. Analysis of the media planning distribution

In this section we analyze the specific investment decisions made by the trained agents (i.e., the redistribution of the budget they make in each time window). Specifically and to make the comparison as simple as possible, we visualize the redistribution in each time window. We show the plots for the B-01un and B-002bal instances in Figs. 10 and 11, whereas the plot for the remaining instances can be found in Figs. D.20 to D.24 in Appendix D. These figures show the percentage of the budget which is reduced from some touchpoints and the additional budget assigned to other touchpoints. Therefore, the sum of the values for each time window must be equal to zero.

We can see in Fig. 10 that in the first window the brand agent decides to withdraw a large part of the investment in touchpoint 8, which originally received all the budget; in order to distribute it between touchpoints 6 and 7, which were originally ignored during the whole simulation. In fact, this is a change that the brand agent maintains in all time windows, both in the training and test periods, reducing the investment in touchpoints 0 and 8 to spread it across touchpoints 6 and 7. Recall that in the unbalanced instances (un) the touchpoints with the greatest impact are 6 and 7, followed by 8 (see Table 2 in previous Section 5.1). On the other hand, Figs. 11 and D.22 show that in balanced instances (bal), where all touchpoints have a similar impact, the learning agent tends to spread the investment more evenly. It always keeps a portion for touchpoints 6 and 7 (which are still the most impactful, although narrowly) but also investing in the rest depending on the situation.

The decisions made by the agent within the Service instances (Figs. D.23 and D.24) show more differences depending on the instance parameters than those previously seen in the Bank instances. In the original media planning (Fig. 7), an increasing part of the budget was invested in touchpoint 4, with a significant part also being invested in touchpoint 0 in the first weeks and in touchpoint 5 in the middle weeks. In S-001a instance, the brand agent tends to invest more in touchpoints 2 and 5 (those with the greatest impact) but without ignoring the rest. On the other hand, in S-008b instance, the agent withdraws less investment from touchpoint 4 (which is the one with the greatest impact in this case), although the percentage is still reduced to incorporate part of the budget in other touchpoints, especially touchpoint 6, which has the second highest impact value.

5.5. Comparison with an econometric model-based strategy

This subsection compares the proposed methodology with an alternative method based on an econometric model [7,8]. Econometric modeling is a frequently used approach in media planning that involves the use of statistical techniques such as regression to inference causal relationships between a set of explanatory variables and a variable of interest to forecast. We first present the econometric strategy and later, qualitative and quantitative comparisons are given.

5.5.1. Econometric model-based strategy description

We build an econometric model as a multivariate linear regression [39] that predicts the expected average awareness of the brand of interest from the proportion of the budget invested in each on-line touchpoint. Since we do not have the real data to build this regression model, we will generate synthetic data through a set of simulations of an ABM for one instance. The idea of this methodology is to use the linear regression as the *ground truth* of a search algorithm to look for the best investment distribution for the entire simulation time. Thus, the linear regression acts as the evaluation function of the search algorithm to guide it. It is worth noting that the ABM could serve as well as an evaluation function. However, each evaluation will be more computationally expensive and therefore, the search algorithm will evaluate a smaller number of alternative choices. We formalize the econometric model-based investment strategy as follows:

1. Evaluation function: an econometric model in the form of a multivariate linear regression. The exploratory or independent variables are the proportion of the available budget to invest in each on-line touchpoint, while the dependent variable is the resulting average awareness of the brand of interest for the entire time frame considered. This regression is built from data generated by running ABM simulations.
2. Decision maker (search algorithm): an algorithm that searches the best static investment distribution to apply among the on-line touchpoints for the entire simulation time. Therefore, its search space is formed by all the vectors of the form $[\text{Inv}_1, \dots, \text{Inv}_{|M_{on}|}] \in [0,1]^{|M_{on}|}$ which satisfy that $\sum_{m=1}^{|M_{on}|} \text{Inv}_m = 1$. The search algorithm applied in the study is a random search which generates different random valid choices from a Dirichlet distribution [40], evaluates them using the econometric model and finally chooses the static investment distribution with the highest evaluation value (i.e., the highest expected awareness for the whole period of simulation time).

5.5.2. Methodological comparison

Before performing some experiments to test this alternative method, we should point out the qualitative differences between this econometric model-based strategy and our RL-based proposal. First, the econometric method provides a static rather than a dynamic strategy. Hence, the decision of the method is only based on the initial state of the ABM and it does not allow to take advantage of the feedback given in the middle of the simulation to modify the investment plan. In addition, this econometric model-based strategy requires investing exactly the same proportion of the budget for each touchpoint throughout the entire simulation. On the other hand, this econometric method has an advantage over our proposal: it considers a continuous action space, allowing to choose any distribution of the investment budget among the touchpoints. We should remind that the DDQN algorithm requires a discrete action space and therefore, our proposal limits the actions to decide the touchpoints to invest.

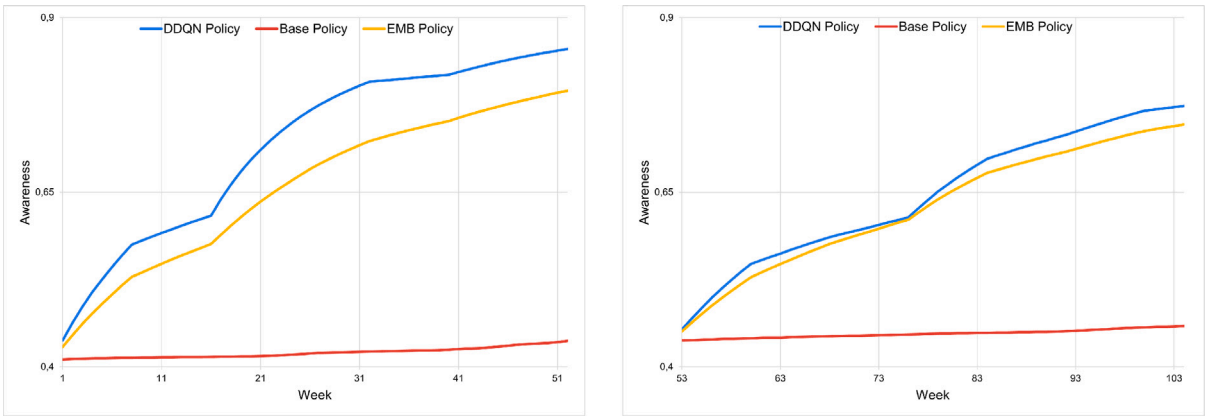
A problem of the econometric method is the bias of the evaluation function: regardless of the quality of the search algorithm, its choices are conditioned by the evaluation function, which is only an approximation of the reality by ABM simulations. Actually, the DDQN method also makes estimates of the quality of each action (the action-value function estimator q_π), but RL provides a more natural framework in which the estimator and the decision maker complement each other to improve their performance at each training iteration step simultaneously.

5.5.3. Comparison of the results for the baseline instance

Here, we carry out some experiments to apply the econometric model-based strategy in the B-original instance, which is the ABM configuration closer to a real scenario. The steps of this experimentation are as follows:

1. We generate 5,000 training samples, each comprising two steps:
   (a) The generation of the input values or features (possible investment distributions). Since the instance B-original have 5 on-line touchpoints, we generate each feature vector from a Dirichlet distribution of 5 variables [40]. This ensures that each feature vector is a valid investment distribution.
   (b) The generation of the output value (the final awareness of the brand for the given investment distribution). To obtain this value requires performing 30 Monte Carlo simulation runs of the training weeks.
2. We build a multivariate linear regression from the previous 5,000 samples. We perform a 5-fold cross-validation in order to determine the quality of the resulting regression, obtaining the mean coefficient of determination R^2.
3. We use the resulting linear regression as the evaluation function for a search algorithm. The algorithm is a random search which considers 100,000 alternative investment distributions.
4. We evaluate this strategy in the same way as our RL-based proposal, by applying its investment choice on the training and test periods separately and analyze the awareness evolution output.

The linear regression (econometric model) obtains an average coefficient of determination in cross-validation of $R^2 = 0.796$. With respect to the search algorithm performance, we show in Fig. 12 its awareness evolution results for both training and test weeks, together with the results of the base and DDQN policies shown previously in Fig. 8. The average awareness of the econometric model-based strategy is 0.657 in the training weeks and 0.629 in the test weeks. Remember that the DDQN policy obtained a mean awareness of 0.718 and 0.645 respectively (see Table 4). Therefore, although the econometric model-based strategy has achieved good results above those of the base policy, they are worse than the results obtained by our RL-based proposed. Results of both strategies in the test weeks are closer. The available budget in test weeks is considerably lower than in training weeks and therefore, it is more difficult to achieve major changes only by readjusting the investment distribution.



(a) Training weeks (b) Test weeks

Fig. 12. Comparison of the awareness output of simulation in B-original instance with respect to the econometric model-based strategy (EMB Policy).

6. Conclusions and future works

The main goal of this work was to design and train a learning agent using RL techniques to obtain a dynamic media planning strategy able to maximize the brand awareness of a business within an ABM built using real market data. For this purpose we chose the value-based method DDQN, trained a brand agent in seven instances of the ABM, and analyzed its results by comparing them with those given by a static media planning strategy previously used by the real brand.

The DDQN brand agent has outperformed the results obtained with the original investment policy, which is based on a static media investment strategy. This improvement is greater in the scenario variants with a larger imbalance in the touchpoint impact values, environments where the correct distribution of the available budget is crucial. We also compared our method with an econometric model-based strategy in the most realistic ABM instance, obtaining higher awareness results as well. In all cases the brand agent was evaluated in a set of test weeks not known during training, where it also outperformed both static investment strategies. Although the improvement in test is tight, it should be noted that in these weeks the available budget is considerably lower than in the training period, which makes more difficult the performance improvement by the brand agent.

The agent does not know the impact of each touchpoint in advance, but it manages to determine which ones are the most interesting in each case. While in unbalanced environments it tends to invest only in the touchpoints with the highest impact, in balanced environments it tends to distribute the investment among several of them, varying its choice according to the specific situation. It is worth noting that the brand agent has achieved the best results in the scenario variant with the parameter setting closer to a real market. This supports the use of the method to build a decision support system to query the best media planning distribution based on the current state of the market.

As future works, we propose to explore other RL techniques such as policy-based methods [41]. We also propose to use the rest of the brands as reactive agents and to consider other KPIs whose values are affected by the action of the competing brands. This competitive scenario would make even more relevant the dynamic behavior of the agent, requiring greater changes in the investment strategy between different time windows than those observed in the current study.

CRediT authorship contribution statement

Víctor A. Vargas-Pérez: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Pablo Mesejo:** Conceptualization, Methodology, Writing – review & editing, Visualization, Supervision. **Manuel Chica:** Conceptualization, Methodology, Investigation, Resources, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Oscar Cordón:** Conceptualization, Methodology, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

This work was supported by the Spanish Ministry of Science, Innovation and Universities, the Andalusian Government, Spain, the University of Granada, Spain, the Spanish National Agency of Research (AEI), and European Regional Development Funds (ERDF), Spain under grants CONFIA (PID2021-122916NB-I00), AIMAR (A-TIC-284-UGR18), and SIMARK (P18-TP-4475). V. Vargas is also supported through the FPU program, Spain (FPU20/02441). M. Chica is also supported through the Ramón *y* Cajal program, Spain (RYC-2016-19800).

Appendix A. Hyperparameter tuning results

See Table A.5.

Appendix B. Return evolution over training

See Figs. B.13 and B.14.

Appendix C. Awareness evolution of trained agent

See Figs. C.15–C.19.

Appendix D. Alteration of the investment distribution

See Figs. D.20–D.24.

Table A.5
Tuning results with different hyperparameter settings in B-002un and B-01bal instances. The score indicates the normalized average test gain with respect to the rest of the average gains of the same instance. The final score is the average score on both instances. The results are ordered with respect to the final score. The best average gain and score values are highlighted in bold.

W	Batch size	Replay memory size	γ	B-002un avg. test gain	B-002un score	B-01bal avg. test gain	B-01bal score	Final score
8	64	20K	0.95	0.1486	0.614	**0.066**	**1.3012**	**0.9576**
8	16	20K	0.99	0.1483	0.5644	0.0658	1.1904	0.8774
8	16	10K	0.99	0.148	0.5147	0.0657	1.135	0.8249
8	64	20K	0.99	0.148	0.5147	0.0657	1.135	0.8249
8	64	10K	0.9	0.1483	0.5644	0.0655	1.0243	0.7943
6	32	30K	0.9	0.1481	0.5313	0.0655	1.0243	0.7778
8	32	20K	0.9	0.148	0.5147	0.0655	1.0243	0.7695
8	16	10K	0.95	0.1482	0.5478	0.0654	0.9689	0.7584
8	16	30K	0.95	0.1482	0.5478	0.0654	0.9689	0.7584
8	16	20K	0.95	0.1481	0.5313	0.0654	0.9689	0.7501
8	16	30K	0.99	0.148	0.5147	0.0654	0.9689	0.7418
8	32	10K	0.9	0.1476	0.4485	0.0654	0.9689	0.7087
8	64	30K	0.95	0.1459	0.1671	0.0659	1.2458	0.7065
6	16	30K	0.99	0.1489	0.6637	0.065	0.7474	0.7055
4	16	10K	0.95	0.1496	0.7795	0.0646	0.5258	0.6527
8	64	30K	0.9	0.1459	0.1671	0.0657	1.135	0.6511
8	32	20K	0.99	0.1453	0.0678	0.0657	1.135	0.6014
8	64	30K	0.99	0.1452	0.0513	0.0657	1.135	0.5932
6	32	10K	0.95	0.1487	0.6306	0.0646	0.5258	0.5782
8	32	10K	0.99	0.148	0.5147	0.0648	0.6366	0.5757
8	32	30K	0.95	0.1443	−0.0977	0.0658	1.1904	0.5464
8	64	20K	0.9	0.145	0.0182	0.0654	0.9689	0.4935
8	16	20K	0.9	0.143	−0.3128	0.0659	1.2458	0.4665
6	16	10K	0.99	0.1493	0.7299	0.0639	0.1381	0.434
8	64	10K	0.99	0.1428	−0.3459	0.0658	1.1904	0.4222
6	64	10K	0.99	0.1493	0.7299	0.0638	0.0827	0.4063
6	16	20K	0.99	0.1488	0.6471	0.0639	0.1381	0.3926
6	64	20K	0.99	0.1492	0.7133	0.0637	0.0274	0.3703
6	32	20K	0.9	0.1484	0.5809	0.0638	0.0827	0.3318
8	32	30K	0.99	0.1487	0.6306	0.0637	0.0274	0.329
6	64	30K	0.99	0.1483	0.5644	0.0638	0.0827	0.3236
4	16	20K	0.9	0.1445	−0.0646	0.0649	0.692	0.3137
6	32	10K	0.99	0.1467	0.2996	0.0641	0.2489	0.2742
8	64	10K	0.95	0.1412	−0.6108	0.0657	1.135	0.2621
6	32	30K	0.95	0.1411	−0.6273	0.0657	1.135	0.2539
6	64	20K	0.95	0.1453	0.0678	0.0642	0.3043	0.1861
4	64	10K	0.95	0.1448	−0.0149	0.0643	0.3597	0.1724
4	16	20K	0.95	**0.1504**	**0.912**	0.0625	−0.6373	0.1373
8	16	30K	0.9	0.1393	−0.9252	0.0658	1.1904	0.1326
8	32	20K	0.95	0.1431	−0.2963	0.0646	0.5258	0.1148
6	32	20K	0.95	0.1438	−0.1804	0.0643	0.3597	0.0896
4	32	10K	0.9	0.1445	−0.0646	0.064	0.1935	0.0645
4	16	30K	0.95	**0.1504**	**0.912**	0.0622	−0.8034	0.0543
4	16	30K	0.99	**0.1504**	**0.912**	0.0622	−0.8034	0.0543
6	64	10K	0.95	0.1488	0.6471	0.0626	−0.5819	0.0326
8	32	10K	0.95	0.1391	−0.9583	0.0654	0.9689	0.0053
4	16	30K	0.9	0.144	−0.1473	0.0639	0.1381	−0.0046
8	16	10K	0.9	0.1483	0.5644	0.0625	−0.6373	−0.0364
8	32	30K	0.9	0.1481	0.5313	0.0625	−0.6373	−0.053
4	16	10K	0.99	0.1421	−0.4618	0.0642	0.3043	−0.0788
6	32	20K	0.99	0.1487	0.6306	0.0622	−0.8034	−0.0864
6	16	10K	0.95	0.1489	0.6637	0.0621	−0.8588	−0.0976
4	32	20K	0.9	0.1444	−0.0811	0.0634	−0.1388	−0.11
6	16	20K	0.95	0.1484	0.5809	0.0622	−0.8034	−0.1112
6	64	30K	0.9	0.1493	0.7299	0.0618	−1.025	−0.1475
4	32	10K	0.95	0.1417	−0.528	0.064	0.1935	−0.1673
4	64	20K	0.95	0.143	−0.3128	0.0636	−0.028	−0.1704
4	64	20K	0.99	0.1419	−0.4949	0.0637	0.0274	−0.2338
4	64	30K	0.9	0.1402	−0.7763	0.0642	0.3043	−0.236
6	16	30K	0.9	0.1489	0.6637	0.0613	−1.3019	−0.3191
6	32	10K	0.9	0.1411	−0.6273	0.0636	−0.028	−0.3277
4	32	10K	0.99	0.1427	−0.3625	0.063	−0.3603	−0.3614
4	32	30K	0.9	0.1449	0.0016	0.0622	−0.8034	−0.4009
6	32	30K	0.99	0.1467	0.2996	0.0616	−1.1357	−0.4181
4	32	30K	0.99	0.1449	0.0016	0.0621	−0.8588	−0.4286
4	16	20K	0.99	0.1408	−0.677	0.0633	−0.1942	−0.4356
6	16	10K	0.9	0.1488	0.6471	0.0609	−1.5234	−0.4381
4	64	20K	0.9	0.1394	−0.9087	0.0632	−0.2496	−0.5791
6	16	20K	0.9	0.1489	0.6637	0.0603	−1.8557	−0.596
6	16	30K	0.95	0.1489	0.6637	0.0603	−1.8557	−0.596

(*continued on next page*)

Table A.5 (*continued*).

W	Batch size	Replay memory size	γ	B-002un avg. test gain	B-002un score	B-01bal avg. test gain	B-01bal score	Final score
4	64	10K	0.9	0.1427	−0.3625	0.062	−0.9142	−0.6383
4	64	30K	0.95	0.1391	−0.9583	0.0629	−0.4157	−0.687
4	64	30K	0.99	0.1497	0.7961	0.059	−2.5757	−0.8898
4	64	10K	0.99	0.1462	0.2168	0.06	−2.0219	−0.9025
6	64	10K	0.9	0.1392	−0.9418	0.0609	−1.5234	−1.2326
4	32	20K	0.95	0.1378	−1.1735	0.0613	−1.3019	−1.2377
6	64	20K	0.9	0.1467	0.2996	0.0586	−2.7973	−1.2489
4	16	10K	0.9	0.1365	−1.3887	0.0614	−1.2465	−1.3176
4	32	30K	0.95	0.1295	−2.5473	0.0631	−0.305	−1.4261
4	32	20K	0.99	0.1324	−2.0673	0.0607	−1.6342	−1.8507
6	64	30K	0.95	0.1059	−6.4533	0.063	−0.3603	−3.4068

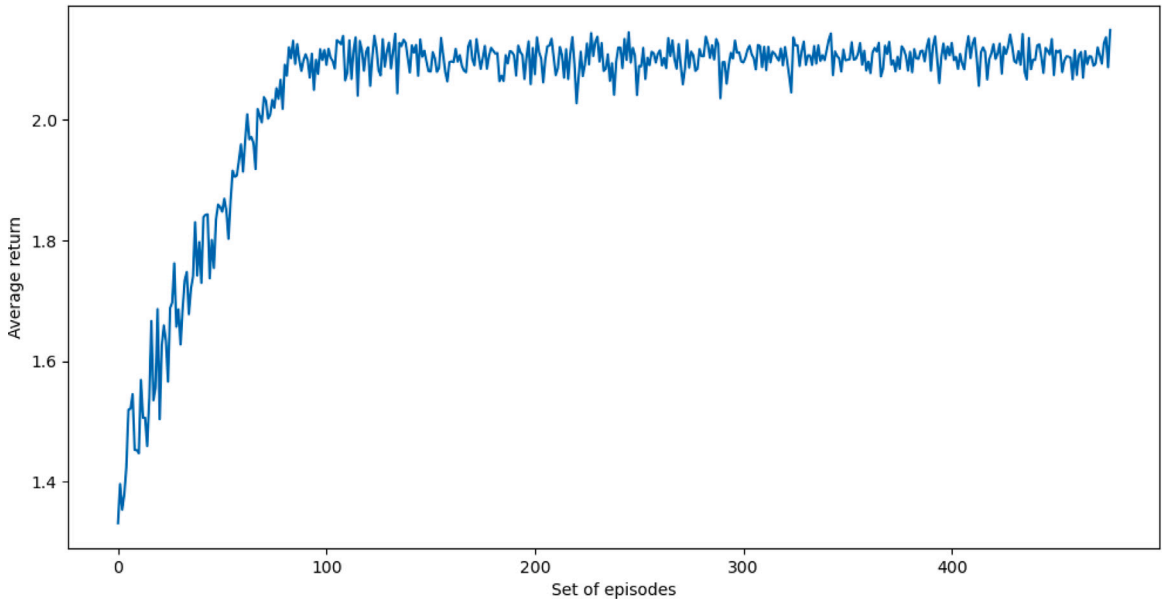


Fig. B.13. Return evolution over training episodes in B-original instance. Each value is the average return of 30 consecutive episodes.

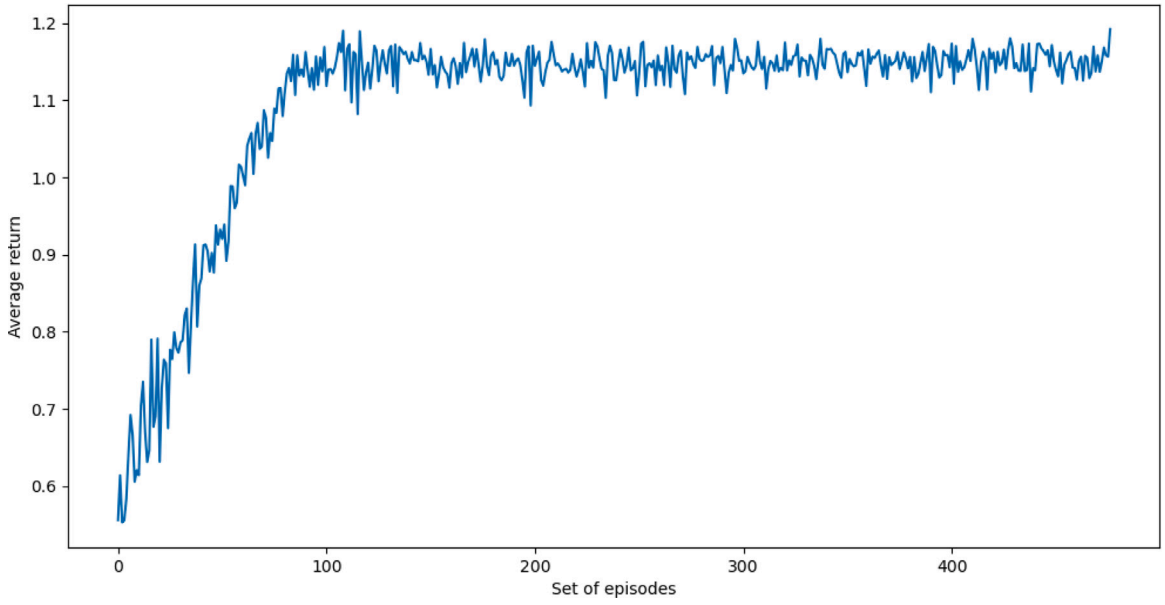
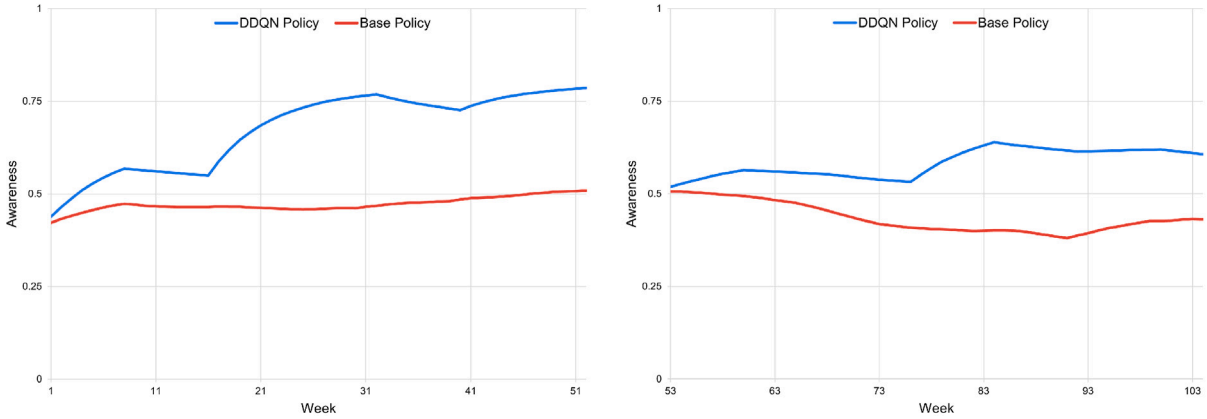
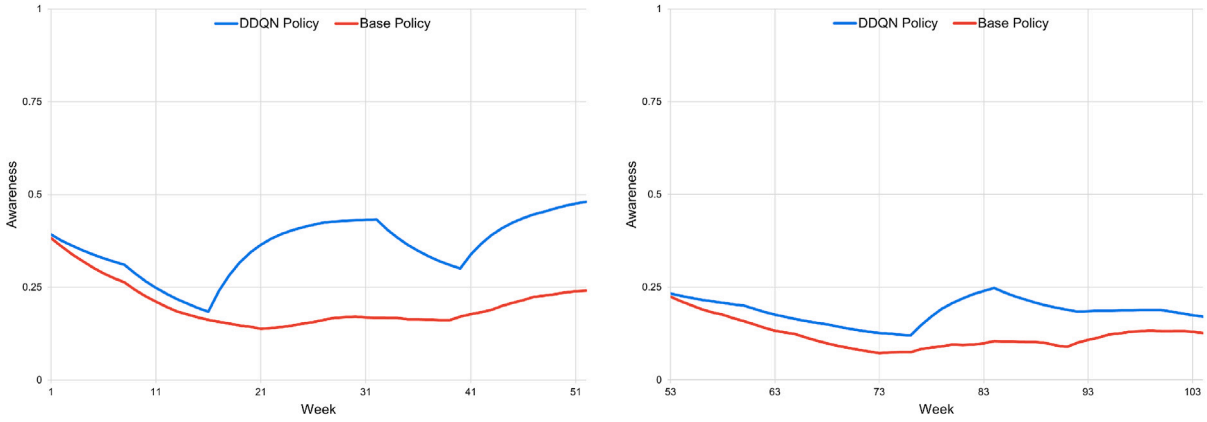


Fig. B.14. Return evolution over training episodes in B-002bal instance. Each value is the average return of 30 consecutive episodes.



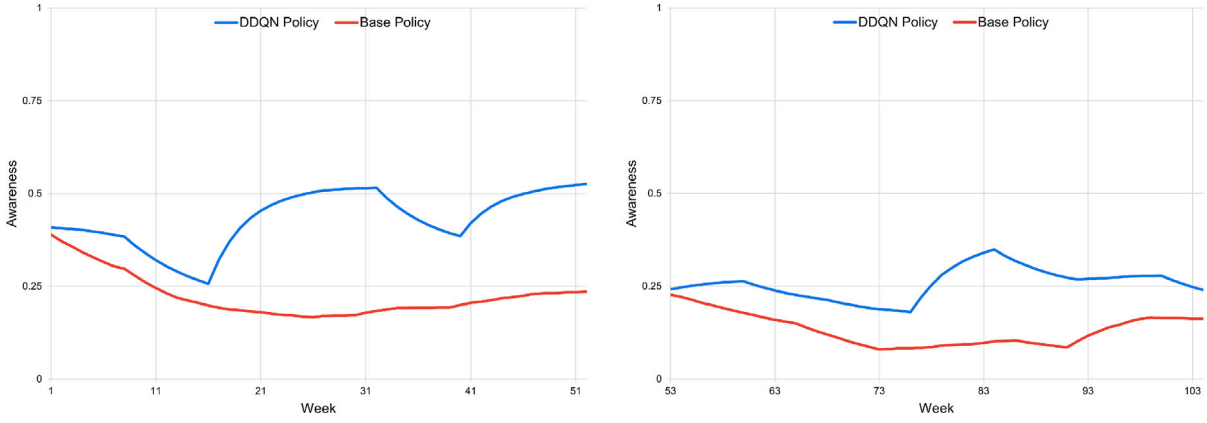
(a) Training weeks (b) Test weeks

Fig. C.15. Awareness output of simulation in B-002un instance.



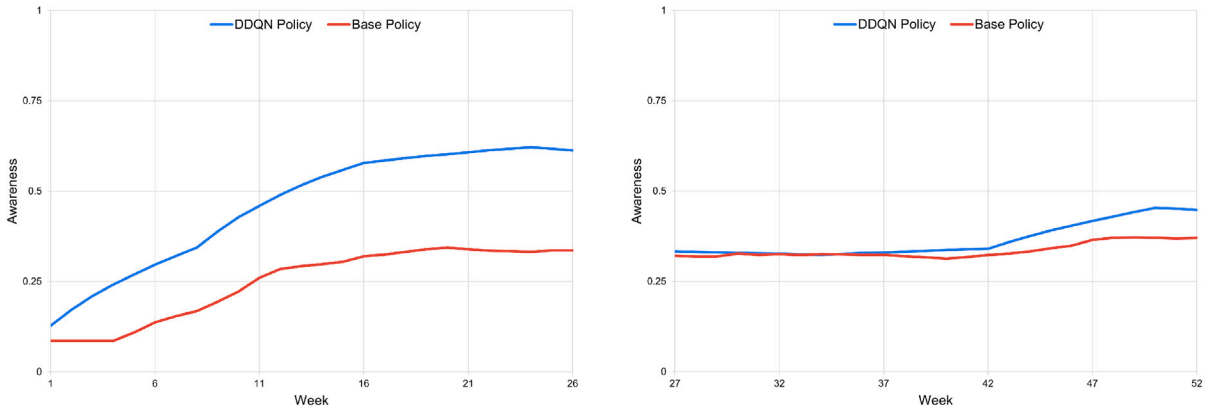
(a) Training weeks (b) Test weeks

Fig. C.16. Awareness output of simulation in B-01bal instance.



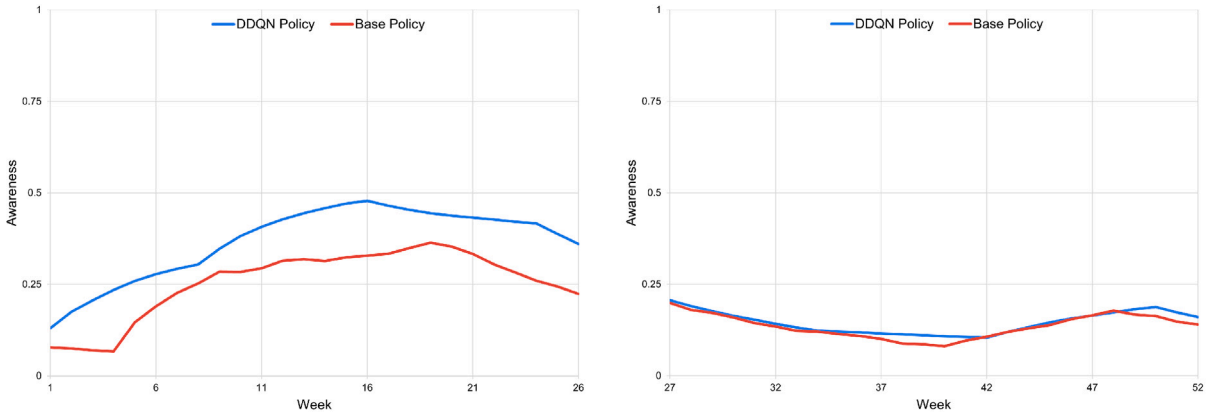
(a) Training weeks (b) Test weeks

Fig. C.17. Awareness output of simulation in B-01un instance.



(a) Training weeks (b) Test weeks

Fig. C.18. Awareness output of simulation in S-001a instance.



(a) Training weeks (b) Test weeks

Fig. C.19. Awareness output of simulation in S-008b instance.

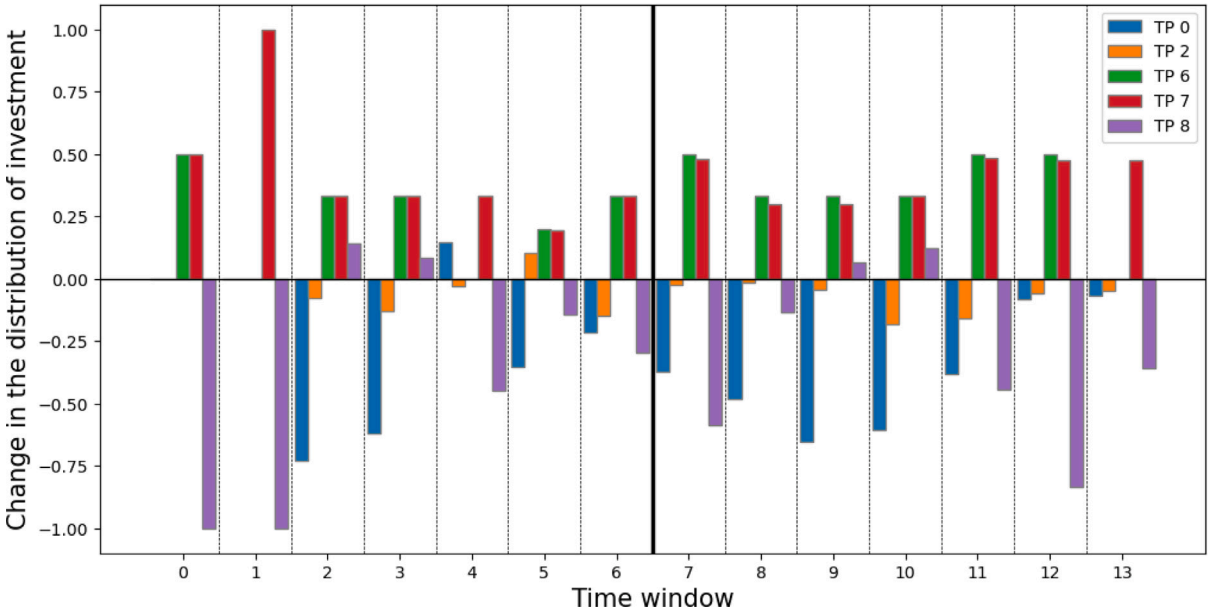


Fig. D.20. Alteration of the investment distribution among the touchpoints (TP) by the brand agent in the B-original instance. The vertical black line separates the training and test windows.

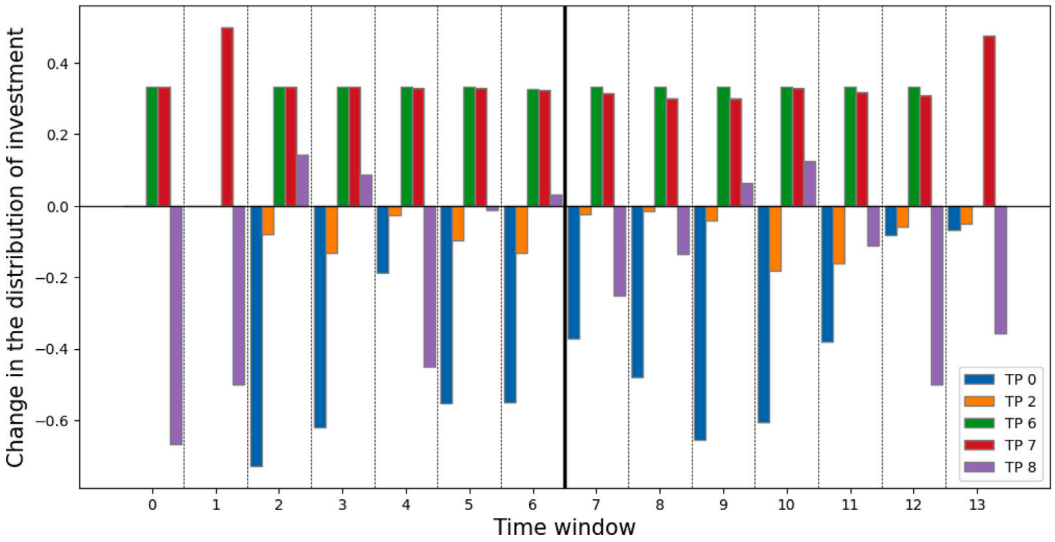


Fig. D.21. Alteration of the investment distribution among the touchpoints (TP) by the brand agent in the B-002un instance. The vertical black line separates the training and test windows.

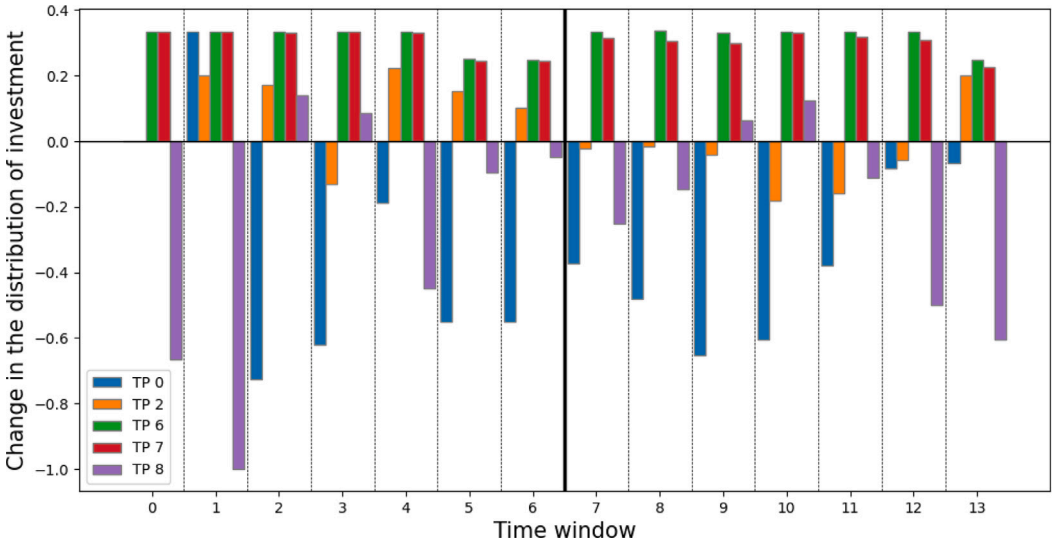


Fig. D.22. Alteration of the investment distribution among the touchpoints (TP) by the brand agent in the B-01bal instance. The vertical black line separates the training and test windows.

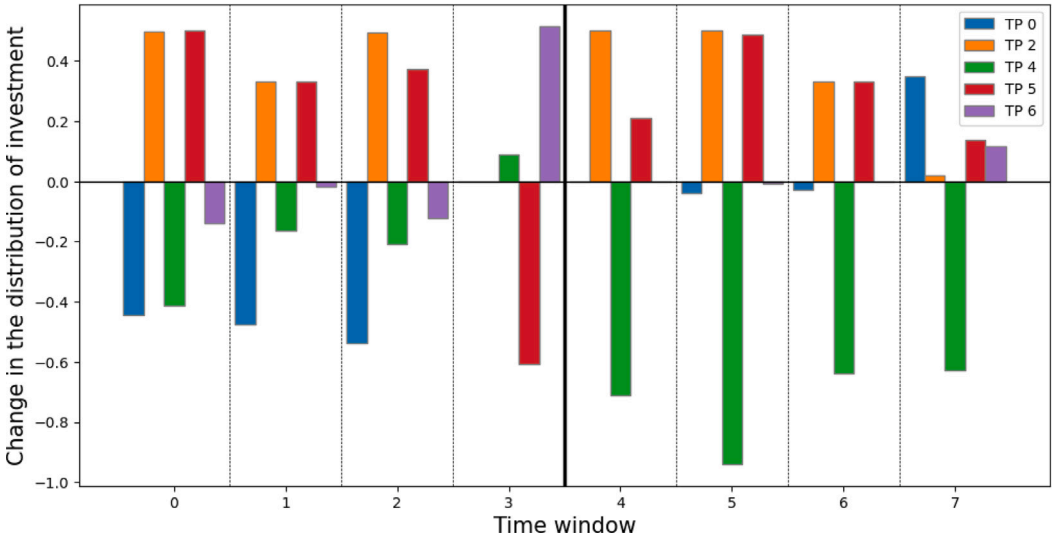


Fig. D.23. Alteration of the investment distribution among the touchpoints (TP) by the brand agent in the S-001a instance. The vertical black line separates the training and test windows.

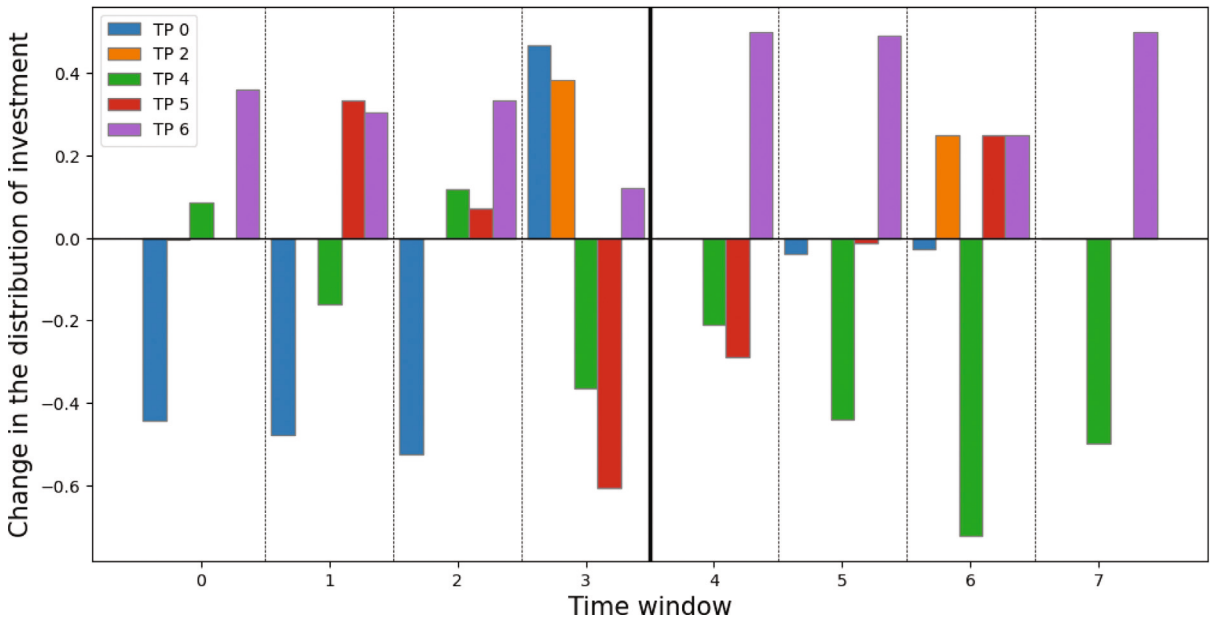


Fig. D.24. Alteration of the investment distribution among the touchpoints (TP) by the brand agent in the S-008b instance. The vertical black line separates the training and test windows.

References

[1] E. Bonabeau, Agent-based modeling: Methods and techniques for simulating human systems, Proc. Natl. Acad. Sci. USA 99 (suppl 3) (2002) 7280–7287.

[2] M. Chica, W. Rand, Building agent-based decision support systems for word-of-mouth programs: a freemium application, J. Mark. Res. 54 (5) (2017) 752–767.

[3] S. van der Hoog, Surrogate modelling in (and of) agent-based models: A prospectus, Comput. Econ. 53 (3) (2019) 1245–1263.

[4] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.

[5] I. Moya, E. Bermejo, M. Chica, Ó. Cordón, Coral reefs optimization algorithms for agent-based model calibration, Eng. Appl. Artif. Intell. 100 (2021) 104170.

[6] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI, vol. 30, AAAI Press, 2016, pp. 2094–2100.

[7] K. Pauwels, I. Currim, M.G. Dekimpe, D.M. Hanssens, N. Mizik, E. Ghysels, P. Naik, Modeling marketing dynamics by time series econometrics, Mark. Lett. 15 (4) (2004) 167–183.

[8] J. Dawes, R. Kennedy, K. Green, B. Sharp, Forecasting advertising and media effects on sales: Econometrics and alternatives, Int. J. Mark. Res. 60 (6) (2018) 611–620.

[9] M. Mitchell, Complexity - a Guided Tour, Oxford University Press, ISBN: 978-0-19-512441-5, 2009, URL http://ukcatalogue.oup.com/product/9780195124415.do.

[10] M.E.J. Newman, The Structure and Dynamics of Networks, in: Princeton studies in complexity, Princeton University Press, ISBN: 9780691113579, 2006, URL http://www.worldcat.org/oclc/62308152.

[11] W. Rand, R.T. Rust, Agent-based modeling in marketing: Guidelines for rigor, Int. J. Res. Mark. 28 (3) (2011) 181–193.

[12] A.J. McLane, C. Semeniuk, G.J. McDermid, D.J. Marceau, The role of agent-based models in wildlife ecology and management, Ecol. Modell. 222 (8) (2011) 1544–1556.

[13] W. Rand, J. Herrmann, B. Schein, N. Vodopivec, An agent-based model of urgent diffusion in social media, J. Artif. Soc. Soc. Simul. 18 (2) (2015) 1.

[14] S.L. Chang, N. Harding, C. Zachreson, O.M. Cliff, M. Prokopenko, Modelling transmission and control of the COVID-19 pandemic in Australia, Nature Commun. 11 (1) (2020) 1–13.

[15] I. Moya, M. Chica, J.L. Saez-Lozano, O. Cordon, An agent-based model for understanding the influence of the 11-M terrorist attacks on the 2004 Spanish elections, Knowl. Based. Syst. 123 (2017) 200–216.

[16] E. Alvarez, J.G. Brida, An agent-based model of tourism destinations choice, Int. J. Tour. Res. 21 (2) (2019) 145–155.

[17] H. Zhang, T. Yu, Taxonomy of reinforcement learning algorithms, in: Deep Reinforcement Learning, Springer, 2020, pp. 125–133.

[18] R.S. Sutton, Learning to predict by the methods of temporal differences, Mach. Learn. 3 (1) (1988) 9–44.

[19] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv preprint arXiv:1312.5602.

[21] P. Ladosz, L. Weng, M. Kim, H. Oh, Exploration in deep reinforcement learning: A survey, Inf. Fusion 85 (2022) 1–22.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[23] H. Hasselt, Double Q-learning, Adv. Neural Inf. Process. Syst. 23 (2010) 2613–2621.

[24] H. Peng, Y. Ma, S. Poria, Y. Li, E. Cambria, Phonetic-enriched text representation for Chinese sentiment analysis with reinforcement learning, Inf. Fusion 70 (2021) 88–99.

[25] Y. Birman, S. Hindi, G. Katz, A. Shabtai, Cost-effective ensemble models selection using deep reinforcement learning, Inf. Fusion 77 (2022) 133–148.

[26] E.F. Domingos, J. Grujić, J.C. Burguillo, F.C. Santos, T. Lenaerts, Modeling behavioral experiments on uncertainty and cooperation with population-based reinforcement learning, Simul. Model. Pract. Theory 109 (2021) 102299.

[27] G. Jäger, Replacing rules by neural networks a framework for agent-based modelling, Big Data Cogn. Comput. 3 (4) (2019) 51.

[28] A. Collins, J. Sokolowski, C. Banks, Applying reinforcement learning to an insurgency agent-based simulation, J. Def. Model. Simul. 11 (4) (2014) 353–364.

[29] H. Hou, T. Gan, Y. Yang, X. Zhu, S. Liu, W. Guo, J. Hao, Using deep reinforcement learning to speed up collective cell migration, BMC Bioinform. 20 (18) (2019) 1–10.

[30] E. Sert, Y. Bar-Yam, A.J. Morales, Segregation dynamics with reinforcement learning and agent based modeling, Sci. Rep. 10 (1) (2020) 1–12.

[31] J.Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, T. Graepel, Multi-agent reinforcement learning in sequential social dilemmas, 2017, arXiv preprint arXiv:1702.03037.

[32] K. Schmid, L. Belzner, T. Gabor, T. Phan, Action markets in deep multi-agent reinforcement learning, in: ICANN, Springer, 2018, pp. 240–249.

[33] Y. Zhu, D. Simester, J.A. Parker, A. Schoar, Dynamic marketing policies: Constructing Markov states for reinforcement learning, SSRN Electron. J. (2020).

[34] S. Tian, S. Mo, L. Wang, Z. Peng, Deep reinforcement learning-based approach to tackle topic-aware influence maximization, Data Sci. Eng. 5 (1) (2020) 1–11.

[35] I. Moya, M. Chica, Ó. Cordón, A multicriteria integral framework for agent-based model calibration using evolutionary multiobjective optimization and network-based visualization, Decis. Support Syst. 124 (2019) 113111.

[36] P. Erdös, A. Rényi, On random graphs i, Publ. Math. 6 (1959) 290–297.

[37] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.

[38] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2015.

[39] B.G. Tabachnick, L.S. Fidell, J.B. Ullman, Using Multivariate Statistics, vol. 5, Pearson Boston, MA, 2007.

[40] T. Minka, Estimating a Dirichlet distribution, Technical report, MIT, 2000.

[41] R.S. Sutton, D.A. McAllester, S.P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in Neural Information Processing Systems, The MIT Press, 2000, pp. 1057–1063.