

Outbound Traffic Control in Call/Contact Center Systems Using the TD3 Algorithm

Marcin Kozłowski
0009-0003-1540-4753
Kielce University
of Technology
Faculty of Electrical
Engineering, Automatic
Control and Computer
Science, Kielce, Poland
Email:
m.kozlowski@tu.kielce.pl

Mirosław Plaza
0000-0001-9728-3630
Kielce University
of Technology
Faculty of Electrical
Engineering, Automatic
Control and Computer
Science, Kielce, Poland
Email:
m.plaza@tu.kielce.pl

Małgorzata Lucińska
0000-0002-7531-0260
Kielce University
of Technology
Faculty of Management
and Computer Modelling
Kielce, Poland
Email:
lucinska@tu.kielce.pl

Marcin Kowalczyk
0000-0001-6137-0580
Warsaw University
of Technology
Faculty of Electronics
and Information Technology
Warsaw, Poland
Email:
marcin.kowalczyk@pw.edu.pl

Abstract—Outbound traffic control in Call Center systems requires simultaneous consideration of efficient utilization of human resources and compliance with stringent quality requirements, particularly those related to acceptable call abandonment levels. This paper proposes an adaptive approach based on reinforcement learning, in which the control task is formulated as a continuous control problem. The operational policy is derived using the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm, characterized by a stable learning process and deterministic selection of action. A simulation environment and a reward function were developed to incorporate key operational performance indicators, such as the consultant occupancy rate (OR) and the abandonment rate (AR). The effectiveness of the proposed solution was evaluated using historical data from a real Call Center system, demonstrating improved resource utilization while maintaining a high level of service quality.

Index Terms—reinforcement learning, call/contact center systems, outbound traffic control, TD3 model.

I. INTRODUCTION

ACTIVITIES involving direct interaction with customers are often entrusted to specialized entities referred to as Call/Contact Center (CC) systems. In their operations, these systems employ a range of advanced solutions designed to address increasing market demands. Such technologies include automatic speech transcription [1], virtual assistants [2], real-time emotion analysis [3]. Many of these solutions are fundamentally based on machine learning and artificial intelligence techniques. They are applied in the handling of both inbound and outbound traffic.

Focusing, in accordance with the scope of this paper, exclusively on outbound traffic, it should be noted that the operation of outbound campaigns takes place in an environment characterized by high operational variability. This variability results primarily from continuous fluctuations in the

number of calls answered by customers, as well as from changes in consultant availability occurring throughout the working day. Under such conditions, overly aggressive control of the number of initiated calls leads to generating more contacts than can be handled by the available staff. Consequently, some customers, despite answering the call, are not connected to a consultant within an acceptable time frame.

In this context, the problem of outbound traffic control has been increasingly analyzed using data-driven and hybrid approaches that extend classical queueing-based models with learning and optimization capabilities [4]. These methods enable capturing complex relationships between call intensity, customer behavior, and service capacity under non-stationary conditions [5]. However, their practical deployment still faces challenges related to system dynamics and real-time adaptability [6].

This typically results in an increase in one of the key quality performance indicators (KPIs), namely the abandonment rate (AR), which is an undesirable situation [7]–[9]. At the same time, prolonged periods of consultant idle time caused by empty waiting queues are also undesirable, as they negatively affect the value of another key performance indicator, namely the consultant occupancy rate (OR). This leads to the problem of dynamically balancing these indicators through appropriate operational actions within outbound traffic control. Due to the high variability of the operating environment, designing an effective algorithm to manage this process constitutes a non-trivial challenge.

The outlined research problem is illustrated schematically in Fig. 1, which presents the operational structure of a Call/Contact Center (CC) system in the context of outbound traffic. As can be observed, not all calls initiated by the system, denoted by the symbol c , are ultimately completed. The process includes, among others, calls that remain unanswered

by customers, for example due to call rejection or an active voicemail system. In particular, calls are also abandoned by customers as a result of excessive waiting time in the queue before being connected to a consultant, which arises from temporary consultant unavailability caused by an excessive number of simultaneously dialed subscribers.

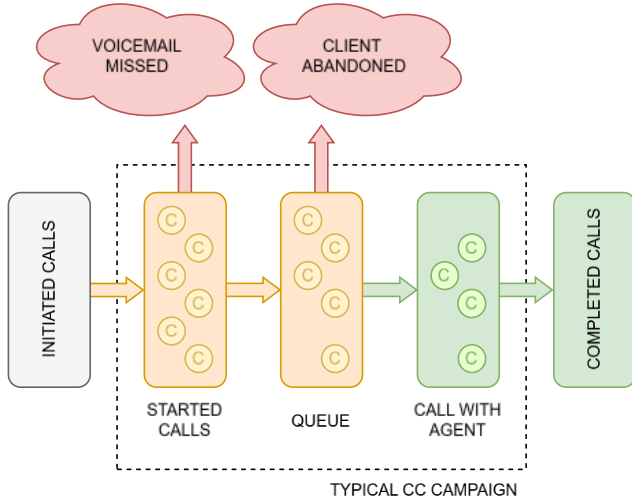


Fig 1. Specifics of contact center operations in the context of outbound traffic

Conversely, overly conservative control limits the number of initiated calls. This leads to underutilization of the consultant workforce and, consequently, to a decrease in OR. As a result, the operational efficiency of the CC system is negatively affected [10].

The presented outbound traffic control problem indicates that maintaining an appropriate balance between AR and OR constitutes a key operational challenge in CC systems. The complexity of this issue, combined with its sensitivity to dynamically changing operating conditions, has led to the development of numerous research approaches [10]–[15], which differ both in their underlying assumptions and in the manner in which the control process is implemented. This challenge is also the focus of the present work. Within this study, an innovative outbound traffic control method based on reinforcement learning (RL) is proposed. In particular, the approach employs the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm to enable continuous, adaptive adjustment of recommendations regarding the number of dialed subscribers, considering the current state of consultant availability as well as the observed responsiveness of contacted customers. To the best of the authors' knowledge, this constitutes the first attempt to apply the TD3 algorithm to adaptive outbound traffic control in CC systems.

The remainder of the paper is organized as follows. Section II provides an analysis of outbound traffic management methods reported in the literature to date, with particular emphasis on their applicability under real operating conditions. Section III focuses on RL, in particular on the actor–critic framework, of which the TD3 algorithm constitutes a representative example. A more detailed conceptual discussion of

this approach is presented in Section IV, while implementation details are described in Section V. The experimental results are reported in Section VI. Finally, Section VII concludes the paper.

II. RELATED WORK

In contrast to inbound traffic, issues related to outbound traffic management in CC systems remained for a long time on the periphery of interest of both academic and business communities. For many years, outbound calls were conducted in a largely random manner. They were directed at broad and unspecialized groups of recipients, without taking customer preferences or contextual information into account. This resulted in low campaign effectiveness and a negative perception of such activities by customers [7], [16], [17].

In recent years, this approach has undergone a significant transformation. Outbound calls increasingly serve informational and supportive functions, including, among others, reminders about appointments, payments, or order status [18]. This has led to greater customer acceptance and contributed to a reduction in unsuccessful call attempts, which, in turn, has a direct impact on improving the operational efficiency of CC systems [19].

Existing approaches to outbound traffic management have focused primarily on three main classes of methods: analytical approaches [10], [11], simulation-based approaches [10]–[12], and heuristic approaches [13]–[15]. Each of these groups differs in terms of underlying assumptions, modeling complexity, and suitability for application under dynamically changing operating conditions of CC systems.

A. Analytical Methods

Analytical methods represent one of the oldest and most well-established approaches used for outbound traffic control in CC systems. These methods rely on mathematical models, in particular queueing models of the M/G/m/m type [11] as well as continuous-time Markov chains (CTMCs) [10]. Such models enable the description of relationships between call initiation intensity, consultant availability, and the risk of an increase in AR. Based on this formulation, it is possible to determine control parameters aimed at reducing consultant idle time while simultaneously meeting quality-of-service requirements. A key advantage of analytical approaches lies in their high structural transparency and the possibility of unambiguous interpretation of operational relationships. However, their main limitation is a strong dependence on assumptions regarding the stability of input parameters. As a result, their effectiveness decreases under conditions of high variability, which are characteristic of real-world CC systems.

B. Simulation-Based Methods

Simulation-based methods complement analytical approaches in situations where the complexity of outbound traffic control makes the use of strict mathematical models infeasible. These methods most commonly rely on discrete-event simulation (DES). They enable realistic modeling of campaign dynamics, customer behavior, and variable consultant

availability [10]. As a result, it becomes possible to evaluate the effectiveness of different control strategies under controlled experimental conditions [12]. A significant advantage of simulation-based methods is the ability to analyze a wide range of operational scenarios without interfering with the functioning CC system. Their main limitations, however, include high computational cost and the need for precise model calibration. Consequently, these approaches are typically applied at the system design stage rather than during real-time operational control.

C. Heuristic Methods

Heuristic methods comprise approaches in which control decisions are made based on a set of adaptive rules, without relying on complete mathematical models or extensive simulations. Their key advantage lies in the ability to operate in real time and to rapidly adjust the intensity of call initiation to the current operational conditions. This enables stabilization of the AR and OR indicators [13]. A more flexible subclass of heuristic methods is represented by metaheuristics, in particular genetic algorithms, which allow exploration of large campaign parameter spaces under conditions of high uncertainty [14].

Heuristic approaches are also widely applied in commercial solutions, where rule-based mechanisms automatically limit the number of initiated calls in response to an increase in AR [15]. Although heuristics offer high responsiveness and low computational cost, their effectiveness strongly depends on the quality of the designed rules and may be limited in scenarios characterized by high system dynamics.

D. Summary of Existing Approaches

The analysis of outbound traffic control methods presented in Sections A–C shows that existing solutions are based on different modeling paradigms. These approaches differ in their level of formalization as well as in the scope of their operational applicability. Analytical methods offer high transparency and the possibility of formal interpretation; however, they exhibit limited adaptability under non-stationary conditions.

Simulation-based approaches enable realistic analysis of complex outbound traffic scenarios by accounting for variability in customer behavior and consultant availability. It should be noted that only in studies [10], [12], [14] were results obtained using simulations driven by data derived from real operational logs of systems. In the remaining works, results were based on scenario-driven simulations and theoretical models, including queueing models [11] and CTMCs [13], using synthetic data or parameters assumed *a priori*. In addition, scenario-based tests were carried out without the use of real operational logs [15].

The comparison of AR and OR values presented in Table I indicates a high variability in the reported results. The comparability of the methods analyzed is further limited by the lack of consistent reporting of the number of consultants. In some studies [10], [15], this parameter is fixed, while in oth-

ers it varies [11], [13], [14], and in certain cases it is not reported at all [12]. This hinders the assessment of the stability of solution effectiveness across a wide range of system configurations and justifies the search for more adaptive traffic control methods.

TABLE I.
PERFORMANCE COMPARISON OF OUTBOUND TRAFFIC CONTROL METHODS

Method	AR (%)	OR(%)	No. of Consultants	Lit.
M/G/m/m	2-5	63-94	0-200	[11]
CTMC	4.0	70.0	6	[10]
DES	N/A	92.0	6	[10]
DES	3-5	85-92	N/A	[12]
AH ¹	1.00-4.78	92.45-93.33	6-10	[13]
GA ²	0.50-2.30	71-78	10-50	[14]
AH-CS ³	0.03-2.95	84.17-97.39	10, 50	[15]

¹ Adaptive Heuristics, ² Genetic Algorithms,

³ Adaptive Heuristics in a Commercial System

It is also worth noting that among the analyzed approaches, only the methods presented in studies [14] and [15] meet the regulatory requirements concerning acceptable levels of AR. These regulations, introduced for example by Ofcom in the United Kingdom (daily limit of 3%) [20] and by the Federal Communications Commission (FCC) in the United States (analogous constraints defined monthly) [21], emphasize the importance of controlling this indicator. As a result, maintaining AR and OR at levels compliant with regulatory standards becomes a key element of effective outbound campaign management.

III. ACTOR–CRITIC METHODS FOR ADAPTIVE OUTBOUND TRAFFIC CONTROL

Based on the results presented in Table I, a natural direction for further consideration is the use of methods that enable learning control strategies directly from observations of system behavior. Such capabilities are provided by RL, in which the control policy is shaped through interaction with a dynamic environment [22], [23]. This approach enables optimization of decision sequences over time, which is of particular importance in CC systems, where each action affects future values of the AR and OR indicators.

A key characteristic of the outbound traffic control problem is the continuous nature of control decisions, resulting from the need to regulate the number of initiated calls at short time intervals. Under these conditions, RL methods based solely on value functions, such as Q-learning [24] or Deep Q-Networks (DQN) [25], are not suitable, as they require discretization of the action space, leading to a loss of control precision [26]. Consequently, policy gradient methods constitute a more appropriate choice, as they allow direct optimization of decision strategies in continuous action spaces [22]. It should be noted, however, that classical policy gradient algorithms suffer from high variance of the gradient estimator, which may result in unstable learning processes in environments characterized by highly variable system dynamics.

These requirements are addressed by actor–critic algorithms, which constitute an extension of policy gradient methods in which the process of updating the control policy is supported by a value function estimator acting as a critic [22], [27]. The presence of the critic enables a significant reduction in gradient variance and stabilization of the learning process, while preserving the ability to generate continuous actions. As a result, actor–critic algorithms combine the advantages of policy-based approaches with decision quality evaluation mechanisms characteristic of value-based methods. Consequently, they are particularly well suited for continuous control tasks subject to stringent operational requirements.

In practice, off-policy actor–critic algorithms are of particular importance for the research problem under consideration, including DDPG [28], TD3 [29], and SAC [30]. This is because these methods employ an experience replay buffer, which enables repeated reuse of collected samples and improves both the stability and efficiency of the learning process. The key properties of these algorithms, relevant from the perspective of outbound traffic control in CC systems, are summarized in Table II.

TABLE II.
COMPARISON OF ACTOR–CRITIC ALGORITHMS FOR CONTINUOUS CONTROL

Feature	DDPG	TD3	SAC
Policy type	Deterministic	Deterministic	Stochastic
Learning stability	Moderate	High	High
Action variability	Moderate	Low	High
Overestimation bias	High	Low	Low
Computational cost	Low	Medium	Medium/High
Real-time suitability	Moderate	High	Moderate
Predictability of actions	Good	Very good	Limited

The comparison presented in Table II illustrates differences among actor–critic algorithms in terms of learning stability and the nature of the generated decisions. In this context, DDPG is characterized by low computational cost but exhibits susceptibility to value function instability [28], as also confirmed by comparative studies [31]. In contrast, SAC provides strong exploration capabilities at the expense of increased control variability, resulting from the stochastic nature of its policy [30], [31]. Against this background, TD3 emerges as a more balanced solution. It should be emphasized that TD3 constitutes an extension of the DDPG method, introducing mechanisms such as double critics, delayed policy updates, and target policy smoothing. These enhancements were designed to mitigate overestimation of the value function [29].

As a result, TD3 ensures stable and deterministic behavior during the operational phase. This property is of key importance for outbound traffic control in CC systems, where limiting short-term decision fluctuations and maintaining AR at levels compliant with regulatory requirements are essential [20], [21], [31]. For these reasons, the TD3 algorithm was selected as the foundation for further work on the proposed outbound traffic control method.

IV. TD3 MODEL ARCHITECTURE

To organize the nomenclature used in this section, the symbols and definitions applied in the subsequent description of the model are introduced. Table III presents the symbols used in the remainder of the paper along with their corresponding descriptions.

TABLE III.
LIST OF DESIGNATIONS USED

Symbol	Description
$nAGENTS(k)$	The total number of consultants at time step k (both busy and idle).
$cAGENTS(k)$	The number of consultants engaged in a call at the current time step k .
$sAGENTS(k)$	The number of consultants occupied with post-call script processing at the current time step k .
$cDELAY$	A constant defining the maximum time within which a customer is expected to answer a dialed call; set to 25 seconds.
$nCALLS(k)$	The number of calls initiated at time step k , resulting from the control decision made at time step $k - 1$.
$cCALLS(k)$	The number of calls answered by customers at the current time step k .
$aCALLS(k)$	The number of calls that were initiated and successfully handled by a consultant at the current time step k .
$autCALLS(k)$	The number of automated calls, i.e., calls that were dialed but not answered by the customer before the time threshold $cDELAY$ at the current time step k .
$cOR(k)$	The percentage of consultants engaged in calls at time step k , defined as: $cOR(k) = \frac{cAGENTS(k)}{nAGENTS(k)} \cdot 100\%$.
$sOR(k)$	The percentage of consultants occupied with post-call script handling at time step k , defined as: $sOR(k) = \frac{sAGENTS(k)}{nAGENTS(k)} \cdot 100\%$.
$AA(k)$	The percentage share of automated calls relative to the number of available consultants at time step k , defined as: $AA(k) = \frac{autCALLS(k)}{nAGENTS(k)} \cdot 100\%$.
$HR(k)$	The call handling success rate at time step k , defined as: $HR(k) = \frac{cCALLS(k)}{nCALLS(k)} \cdot 100\%$.
$AR(k)$	The abandonment rate indicating service quality at time step k , defined as: $AR(k) = \frac{cCALLS(k) - aCALLS(k)}{cCALLS(k)} \cdot 100\%$.
SL	The percentage of calls handled within a predefined waiting time threshold of two seconds.
$avgCALLS$	The normalized average call duration.
$avgSCRIPTS$	The normalized average duration of post-call script processing.
π	The action policy; a rule defining how actions are selected by the RL agent.
ϕ	The parameters of the policy π , i.e., the weights of the neural network model.
$Q^{\pi\phi}$	The action-value function associated with the policy π_{ϕ} .

The proposed framework consists of two main components: a data-driven CC system simulator, which includes an operational layer representing the dynamics of the Call Center (CC) environment, and a control layer based on a TD3 RL agent. Their interaction is mediated by an observation model that captures both the current system state and its recent temporal dynamics. The overall structure of the system is illustrated in Fig. 2.

As shown in Fig. 2, the operational layer does not constitute an independent component but is an integral part of the CC system simulator, which reproduces the dynamics of the CC environment, including call generation, customer behavior,

and resource utilization. These processes are not generated artificially; instead, they are modeled using a data-driven simulation approach based on historical billing data obtained from a real-world CC system. Within the simulator, information is aggregated from time-ordered billing records, where each record corresponds to a single call instance. The number of initiated calls $nCALLS(k)$ determines how many records are processed at a given time step, thereby directly linking control decisions with the underlying data-driven representation of the environment. In this way, the operational layer reflects realistic system behavior derived from actual historical interactions.

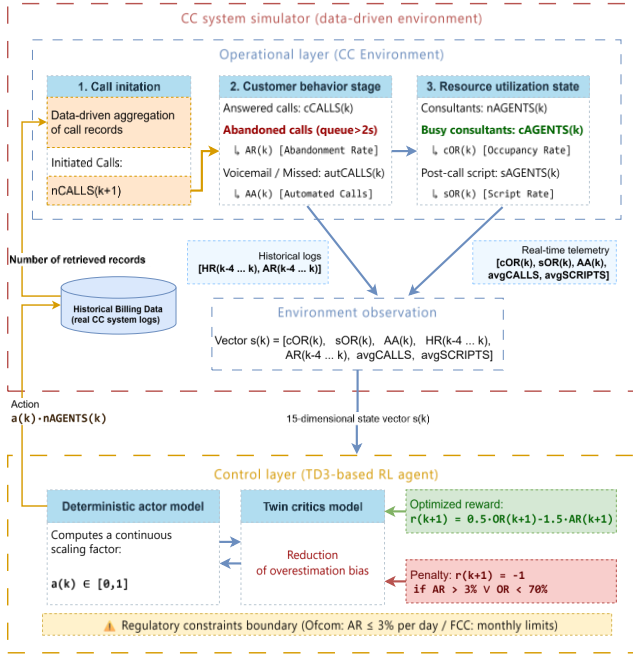


Fig 2. System-level architecture of the proposed RL framework with a data-driven CC system simulator

In this configuration, the control layer does not directly manipulate the environment in a deterministic or scripted manner but interacts with the simulator through control decisions that influence the number of initiated calls and, consequently, the number of active consultants engaged in communication. These decisions are based on a state representation that integrates real-time metrics with short-term historical information, enabling the agent to account for both the current conditions and recent system dynamics.

The agent optimizes its policy by maximizing a reward signal that reflects a trade-off between service efficiency and quality of service, in particular by promoting high consultant occupancy while penalizing excessive call abandonment.

Detailed definitions of individual components, including the state representation and reward formulation, are provided in the following part of this section.

Since the control layer is realized using a RL agent, it is essential to describe the internal architecture and learning mechanism of the applied TD3 algorithm in greater detail. This allows the transition from the system-level perspective

introduced above to the algorithmic level, where the decision-making process is formally defined.

The architecture of the applied TD3 algorithm was designed to support adaptive control of the number of outbound calls in an environment characterized by high variability and regulatory constraints [29]. The schematic of this architecture is presented in Fig. 3. The figure illustrates the complete decision-making cycle of the algorithm, starting from observation of the environment state $s(k)$, through action generation $a(k)$, estimation of the action value, network parameter updates, and finally the feedback signal received from the environment. The individual components are labeled according to the adopted convention: ϕ denotes the parameters of the actor network, θ_1 and θ_2 represent the parameters of the critic networks, and ε denotes Gaussian noise used to stabilize the exploration process.

A key component of the architecture is a learning agent that continuously learns through interaction with the environment. The state of the environment is modeled using a dedicated CC system simulator. At each time step, the agent receives an observation vector $s(k)$. The definition of this vector is based on 15 variables, whose meanings are summarized in Table III. The observation vector takes the following form:

$$s(k) = [cOR(k), sOR(k), AA(k), HR(k-4), HR(k-3), \dots, HR(k-2), HR(k-1), HR(k), AR(k-4), AR(k-3), \dots, AR(k-2), AR(k-1), AR(k), avgCALLS, avgSCRIPTS] \quad (1)$$

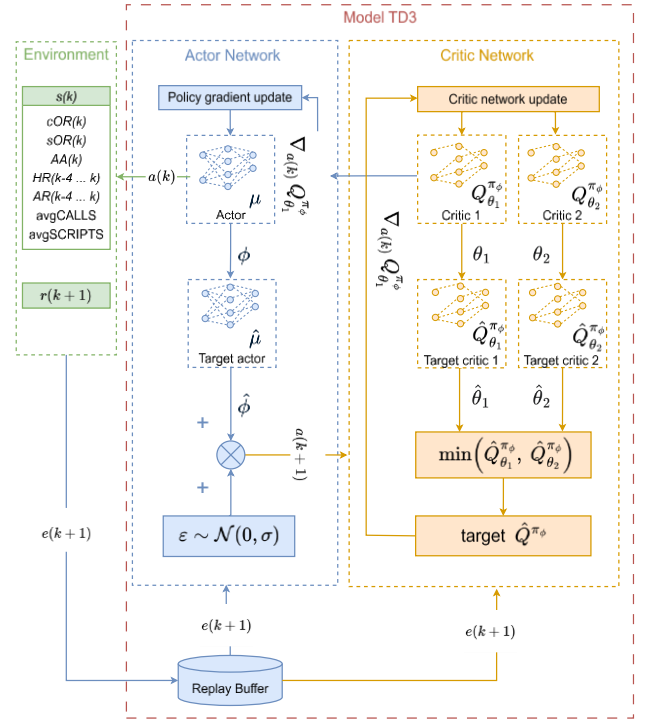


Fig 3. Diagram of the TD3 model architecture

Vector (1) comprises the current operational parameters of the Call/Contact Center system as well as historical values of the HR and AR indicators, in accordance with the adopted five-step observation horizon. Incorporating such a short-term temporal horizon enables the agent to capture the

dynamics of changes in the environment and to anticipate the effects of the decisions being made.

Both the length of the time step and the number of past time steps included in the observations are determined by the specific characteristics of outbound traffic. In the proposed solution, a time step k of five seconds is assumed, reflecting the practical intervals at which new calls are typically initiated. The inclusion of parameters from five-time steps (the current one and four historical steps) is related to the commonly observed maximum waiting time for a customer to answer a call, denoted as $cDELAY$, which is equal to 25 seconds. In accordance with service quality assumptions, it is assumed that a customer abandons the call after a waiting time of two seconds without being connected to a consultant.

Based on the observed environment state represented by vector (1), the actor network, denoted by μ , determines a deterministic operational decision $a(k)$. This relationship, expressed in the context of the Call/Contact Center system, is described by Eqs. (2) and (3) [28], [32].

$$a(k) = \mu(s(k) | \phi) \quad (2)$$

$$[a(k) \cdot nAGENTS(k)] = nCALLS(k+1) \quad (3)$$

The expression $\mu(s(k) | \phi)$ in Eq. (2) denotes the deterministic actor policy function, which maps the current environment state $s(k)$ to a control action. The generated action $a(k)$ is then applied to the environment, which—according to the definition of a single experience $e(k+1)$ given in Eq. (4)—returns a pair $(r(k+1), s(k+1))$ [23].

$$e(k+1) = (s(k), a(k), r(k+1), s(k+1)) \quad (4)$$

The term $r(k+1)$ in Eq. (4) denotes the reward value obtained after executing action $a(k)$ in state $s(k)$, reflecting the quality of the applied control decision. In the considered system, the reward parameters were defined based on service quality requirements derived from the service level (SL). According to this criterion, 98% of calls should be connected within a waiting time not exceeding two seconds. Satisfying this requirement necessitates both a very low AR and a sufficiently high OR. Therefore, the following threshold values were adopted:

$$AR_{max} = 3\%; OR_{min} = 70\%. \quad (5)$$

If any of the thresholds defined in Eq. (5) is violated, the value of the reward function is set to -1 . In all remaining cases, the reward takes the value given by Eq. (6):

$$r(k+1) = \begin{cases} -1, & (AR(k) > AR_{max}) \vee (OR(k) < OR_{min}) \\ w_{OR}OR(k) + w_{AR}AR(k), & (AR(k) \leq AR_{max}) \wedge (OR(k) \geq OR_{min}) \end{cases} \quad (6)$$

where $w_{OR} = 0.5$ and $w_{AR} = -1.5$ denote the weights associated with OR and AR, respectively. The larger absolute value of the weight w_{AR} compared to w_{OR} implies that an increase in AR is penalized more strongly than changes in OR. This design reflects the priority importance of limiting AR. In accordance with applicable regulatory standards and quality requirements for CC systems, AR constitutes a key parameter for assessing service quality [20], [21]. Consequently, the negative value of w_{AR} explicitly favors solutions that minimize AR, even at the expense of less critical changes in OR.

An important component of the TD3 architecture is the replay buffer (see Fig. 3), in which the agent's experiences $e(k+1)$ are stored. This buffer enables random sampling from past interactions, which significantly reduces temporal correlations between successive observations and stabilizes the learning process of the critic networks. By allowing multiple reuses of the same data, sampling efficiency is improved. As a result, the agent learns from a more diverse set of experiences originating from different stages of system operation [28].

To mitigate the problem of systematic overestimation of the action-value function $Q^{\pi\phi}$, the TD3 model employs an architecture with two independent critic networks, denoted as $Q_{\theta_1}^{\pi\phi}$ and $Q_{\theta_2}^{\pi\phi}$. As illustrated in Fig. 3, the target value $\hat{Q}^{\pi\phi}$ is computed as the minimum of the two independent estimates produced by the critics. Such conservative value estimation reduces the risk of propagating overly optimistic action evaluations and stabilizes the update process [29].

Furthermore, to enhance the stability of the learning process, target copies of both the actor network and the two critic networks are employed. The target actor is denoted by $\hat{\mu}$, while the target critics are denoted as $\hat{Q}_{\theta_1}^{\pi\phi}$ and $\hat{Q}_{\theta_2}^{\pi\phi}$. These target networks are used to compute both the target action and the target value of the action-value function $Q^{\pi\phi}$. The target action in state $s(k+1)$ is determined according to Eq. (7) [28], [31].

$$a(k+1) = \hat{\mu}(s(k+1) | \phi) + \varepsilon \quad (7)$$

The added random term ε serves as a mechanism for controlled exploration. The resulting target action is subsequently used by the target critics to compute the target value of the action-value function, denoted as $\hat{Q}^{\pi\phi}$. This value is defined as the sum of the future reward $r(k+1)$, weighted by the discount factor γ , and the minimum of the two available action-value estimates produced by the double-critic architecture. This relationship is expressed by Eq. (8) [29]:

$$\hat{Q}^{\pi\phi}(s(k), a(k)) = r(k+1) + \gamma \cdot \min_{i \in \{1,2\}} Q_{\theta_i}^{\pi\phi}(s(k+1), a(k+1)) \quad (8)$$

Learning of the actor network itself is based on the gradient signal denoted as $\nabla_{a(k)} Q_{\theta_1}^{\pi\phi}(s(k), a(k))$. This signal is computed with respect to the action $a(k)$ by the primary critic $Q_{\theta_1}^{\pi\phi}$. It should be emphasized that, in the TD3 algorithm, updates of the actor parameters are performed in a delayed manner, i.e., not at every simulation step but only after the critic value estimates have sufficiently stabilized. This mechanism limits the influence of transient fluctuations in the value function on the shape of the policy [32].

Another mechanism that stabilizes the learning process is the so-called soft update of the target network parameters. This mechanism consists in gradually updating the parameters of the target networks by blending them with the parameter values of the corresponding main networks. Formally, this relationship can be expressed as [28], [31]:

$$\hat{\omega} \leftarrow \tau \cdot \omega + (1 - \tau) \cdot \hat{\omega}, \quad (9)$$

where ω denotes the parameters of the main network, while $\hat{\omega}$ represents the parameters of the target network. The coefficient $\tau \in (0,1)$ defines the update rate of the parameters. A small value of τ ensures slow and stable propagation of parameter changes, thereby reducing the risk of instability caused by abrupt fluctuations in value estimation. It should be noted that this mechanism is applied both to the actor network and to both critic networks. As a result, the target values used during parameter updates are more smoothly varying and more robust to oscillations [28].

The presented description of the TD3 architecture organizes the key decision-making and learning mechanisms, forming a coherent model for adaptive outbound traffic control. This formulation provides a basis for transitioning from the formal model description to the implementation layer, in which the designed structures must be realized as concrete software solutions.

V. IMPLEMENTATION OF THE TD3 MODEL

The implementation of the proposed outbound traffic control algorithm was carried out in Python. To this end, the Gymnasium library was used to define the environment, while Stable-Baselines3 was employed to provide a reference implementation of the TD3 algorithm. The objective of this section is to focus on the selection of neural network architecture and the key algorithm parameters that determine the stability and effectiveness of the learning process.

A. Action space

In the considered outbound traffic control problem, the agent's decision-making reduces to regulating the intensity of call initiation at successive time steps. The deterministic actor policy determines, at each time step, the action value $a(k)$, which takes the form of a normalized decision signal constrained to the interval $[0,1]$. This value represents the relative level of dialing intensity and does not directly correspond to the absolute number of calls. Only through interaction with the environment is it mapped onto an integer quantity, interpreted as the actual number of calls initiated in the subsequent time step, in accordance with Eqs. (2) and (3).

Such separation of the decision-making layer from the operational layer ensures the physical validity of the model, promotes stability of the learning process, and makes the control policy independent of the absolute scale of the CC system.

B. Neural Network Architecture

The neural network architecture employed in the TD3 algorithm was selected to ensure a balance between approximation capability and computational efficiency [33]. Both the actor network and the critic networks were implemented as a multilayer perceptron, following the reference *MlpPolicy* architecture provided by the Stable-Baselines3 library. The adopted configuration consists of two hidden layers with 256 neurons each [34]. The parameters of both networks are optimized using the Adam algorithm with a constant learning rate to ensure smooth and stable convergence throughout training.

C. Exploration Mechanism

In the proposed implementation, exploration of the action space is achieved by adding additive Gaussian noise to the action generated by the actor during the training phase. This noise is applied exclusively during learning and enables the collection of diverse experiences while preserving the continuous nature of the control process.

The level of exploration is controlled by the standard deviation of the Gaussian distribution, which was set to $\sigma = 0.15$ in the presented solution. This value was selected empirically as a compromise between convergence speed and learning stability. The exploration mechanism can be expressed as:

$$a_{train}(k) = a(k) + \varepsilon(k), \quad \varepsilon(k) \sim \mathcal{N}(0, \sigma^2) \quad (10)$$

where $a(k)$ denotes the action determined by the deterministic actor policy, and $a_{train}(k)$ represents the action executed during the training phase. The adopted value of σ ensures sufficient variability of decisions in the early stages of learning, while simultaneously limiting the risk of excessive fluctuations as the policy stabilizes [29], [35].

The implementation of the described exploration mechanism is illustrated by the code fragment presented in Listing 1. During training, the noise is automatically added to the actions generated by the actor, in accordance with the exploration mechanism implemented in the Stable-Baselines3 library.

```

1  # action dimension
2  action_dim = env.action_space.shape[-1]
3
4  # Gaussian noise configuration for exploration
5  SIGMA = 0.15
6  action_noise = NormalActionNoise(
7      mean=np.zeros(action_dim, dtype=np.float32),
8      sigma=SIGMA * np.ones(action_dim, dtype=np.float32)
9  )

```

Listing 1. Configuration of the exploration noise

D. TD3 Algorithm Configuration

After defining the learning environment and the exploration mechanism, the TD3 algorithm used in the experiments was configured. The configuration primarily includes settings related to the update rate of the target networks, the frequency of critic updates, and the delayed update of the actor network.

In accordance with the description provided in Section B, the *MlpPolicy* architecture was employed, and the configured exploration mechanism presented in Listing 1 was passed to the algorithm. To stabilize the learning process, an experience replay buffer (*ReplayBuffer*) was used, enabling random sampling of collected observations [28], [34].

A key parameter influencing the learning dynamics is the soft update coefficient of the target networks, which was set to $\tau = 0.001$ in the proposed solution. The small value of this parameter ensures slow and controlled propagation of parameter changes, thereby limiting the risk of instability resulting from abrupt fluctuations in critic value estimates [28], [31].

The mini-batch size was set to *batch_size* = 128, representing a compromise between gradient estimator variance

and computational cost, and promoting stable convergence [28], [29], [36].

Critic updates are performed every five environment steps ($train_freq = (5, "step")$), which reduces the impact of data autocorrelation in samples drawn from the replay buffer [28]. In addition, delayed actor updates relative to the critics were applied with $policy_delay = 2$. This mechanism reduces the propagation of errors in estimating the action-value function $Q^{\pi\phi}(s(k), a(k))$ and prevents excessive policy optimization with respect to unstable critic evaluations [29]. All computations were performed using GPU acceleration.

The complete configuration of the TD3 algorithm employed in the experiments is presented in Listing 2.

```

1  model = TD3(
2      policy="MlpPolicy",
3      env=env,
4      action_noise=action_noise,
5      replay_buffer_class=ReplayBuffer,
6      tau=0.001,
7      batch_size=128,
8      train_freq=(5, "step"),
9      policy_delay=2,
10     device="cuda",
11     verbose=1
12 )

```

Listing 2. Initialization of the TD3 model using an MLP policy

E. Policy Evaluation and Model Stopping Criterion

To monitor the quality of the learned policy and reduce the risk of overfitting, an evaluation mechanism based on the *EvalCallback* and *StopTrainingOnRewardThreshold* classes was employed. The evaluation is performed in a deterministic manner, without the inclusion of exploration noise, which allows assessment of the true quality of the control policy [22], [34].

In the analyzed implementation, evaluation is performed once after the completion of the training process, and the obtained result is compared against a predefined operational threshold. The applied quality threshold is set to $reward_threshold = 1000$ and refers to the cumulative reward obtained in a single evaluation episode. Achieving this value indicates that the system maintains correct behavior over a substantial portion of the episode, which makes it possible to distinguish a stable and operationally useful policy from random or accidental solutions [34], [36].

The configuration of the evaluation mechanism and the training stopping criterion is presented in Listing 3.

The $eval_freq$ parameter specifies the number of environment steps between consecutive invocations of the evaluation procedure. In the configuration presented in Listing 3, its value is set equal to the total number of training steps of the model ($total_timesteps = 10000$). As a result, the evaluation is executed only once, after completion of the entire training process. This approach enables assessment of the policy quality in its final, stabilized state, without interrupting the learning process with additional intermediate evaluations [28], [34], [36].

```

1  # Early stopping based on a quality threshold
2
3  callback_on_best = StopTrainingOnRewardThreshold(
4      reward_threshold=1000,
5      verbose=1
6  )
7
8  # Configuration of periodic policy evaluation
9  eval_callback = EvalCallback(
10     eval_env=env,
11     eval_freq=10000,
12     n_eval_episodes=1,
13     deterministic=True,
14     callback_on_new_best=callback_on_best,
15     verbose=1
16 )
17
18 # TD3 model training with evaluation callback
19 model.learn(
20     total_timesteps=10000,
21     callback=eval_callback,
22     log_interval=1
23 )

```

Listing 3. Configuration of the periodic evaluation mechanism

Linking the duration of the training process with the evaluation moment ensures that the quality assessment directly refers to the final, stable version of the policy. When combined with the training stopping criterion, this approach enables automatic termination of training upon reaching the assumed performance level, eliminating the need for further manual monitoring of the optimization process [34], [36].

The parameter $n_eval_episodes = 1$ defines the number of episodes used in a single evaluation session. This setting is sufficient in the analyzed deterministic environment and allows for straightforward interpretation of the obtained cumulative reward [22], [34].

The applied evaluation mechanism and training stopping criterion enable an unambiguous assessment of the final policy quality and automatic termination of the learning process once the desired level of performance has been achieved [36].

VI. EXPERIMENTAL RESULTS

The proposed solution was evaluated using historical data obtained from a real CC system. The reference data, denoted with the prefix R (ROR, RTT, RAR), were collected from a production environment employing a commercial dialer based on adaptive heuristics, which constitutes a competitive industry-standard baseline. The control policy was trained offline using data collected during periods when the system operated with approximately 10 consultants. Subsequently, the learned policy was evaluated in a simulation environment; all results presented in the remainder of this section correspond to a test configuration comprising 24 consultants.

Control decisions regarding the number of initiated calls were made at time steps of five seconds, after which the updated system state was analyzed. The effectiveness of the proposed TD3-based approach was evaluated using two key operational performance indicators, namely AR and OR.

The analysis was conducted on randomly selected two-hour operational periods, exclusively drawn from time intervals during which the actual number of active consultants

exceeded 20. This criterion was adopted to ensure comparability between real data and the simulation configuration, as well as to assess the stability of the control policy under conditions representative of real-world outbound campaigns with moderate to high workload.

The analysis of the data summarized in Table IV indicates a clear increase in the average OR from 78.27% under the adaptive-heuristic-based baseline to 84.33% under the TD3-based policy, corresponding to an absolute increase of 6.06%. At the same time, an increase in the average TT was observed, from 51.50% to 57.22%, i.e., by 5.72%. As a result, the increased workload of consultants translates directly into longer actual interactions with customers rather than merely an increase in auxiliary activity.

The largest relative benefits are observed during periods of initially low occupancy (e.g. $ROR \approx 72\%$), where OR improved by as much as 13%. This confirms effective filling of workload gaps. At the same time, the average AR in the simulation was equal to 0.79%, and in more than 30% of the analyzed time intervals it reached $SAR = 0.00\%$, despite the application of a strict customer patience threshold of two seconds. This shows that the proposed solution satisfies the applicable regulatory requirements regarding acceptable AR levels, including the limits imposed by supervisory authorities such as Ofcom [20] and the FCC [21].

Furthermore, a noticeable reduction in the standard deviation of OR from 2.93% to 1.92% in the simulation indicates greater workload stability compared to the adaptive-heuristic-based baseline.

TABLE IV.

COMPARISON OF CC CONSULTANT TIME UTILIZATION IN PERCENTAGES

Date	Time	ROR	RTT	RAR	SOR	STT	SAR	DOR	DTT	DAR
[yyyy-mm-dd]	[h]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
2021-03-15	10-12	83.14	60.28	2.03	85.79	62.20	1.49	2.65	1.92	0.54
2021-03-15	12-14	79.23	57.44	2.53	85.42	61.93	0.59	6.18	4.48	1.94
2021-03-27	08-10	78.73	48.19	0.91	85.39	57.24	0.00	6.66	9.05	0.91
2021-03-27	10-12	79.19	52.24	1.70	83.92	57.90	0.00	4.73	5.66	1.70
2021-03-29	08-10	78.73	52.77	4.64	85.39	57.24	0.00	6.66	4.47	4.64
2021-03-29	10-12	79.19	54.04	1.22	83.92	57.90	0.00	4.73	3.86	1.22
2021-03-29	12-14	77.48	45.63	2.42	84.62	49.84	0.58	7.13	4.20	1.84
2021-04-21	09-11	79.04	58.50	1.24	85.17	66.04	0.00	6.13	7.53	1.24
2021-04-21	11-13	75.11	50.94	2.02	83.31	58.94	0.00	8.20	8.00	2.02
2021-04-21	13-15	80.40	54.01	2.01	86.27	63.82	0.30	5.87	9.80	1.71
2021-04-21	15-17	84.50	56.16	2.82	86.27	59.07	2.39	1.77	2.91	0.43
2021-04-21	18-20	81.67	54.45	5.30	83.83	57.21	4.36	2.16	2.75	0.94
2021-04-22	10-12	80.08	51.80	2.29	79.42	52.53	0.18	-0.66	0.73	2.11
2021-06-09	14-16	76.24	47.05	1.40	84.90	55.94	0.83	8.66	8.89	0.57
2021-06-09	16-18	71.75	44.67	11.55	85.39	54.52	1.47	13.63	9.85	10.08
2021-06-09	18-20	76.90	44.00	11.04	83.26	48.83	0.00	6.36	4.83	11.04
2021-06-10	08-10	78.20	47.11	0.83	86.58	52.78	3.09	8.38	5.67	-2.26
2021-06-10	10-12	76.74	48.78	1.52	84.99	58.44	0.00	8.25	9.66	1.52
2021-06-10	12-14	76.36	50.55	1.50	85.56	56.11	1.19	9.20	5.56	0.31
2021-06-15	13-15	73.54	45.18	1.14	81.09	52.28	0.20	7.55	7.10	0.94
2021-06-15	15-17	74.45	49.15	0.94	83.31	54.62	0.91	8.87	5.47	0.03
2021-06-23	13-15	77.58	52.16	5.18	79.76	54.58	0.00	2.18	2.42	5.18
2021-06-24	08-10	81.11	56.13	1.69	86.02	61.04	0.37	4.92	4.92	1.32
2021-06-24	10-12	79.18	54.72	1.58	84.31	62.27	1.08	5.13	7.55	0.50
Avg		78.27	51.50	2.90	84.33	57.22	0.79	6.06	5.72	2.10
Std.		2.93	4.57	2.87	1.92	4.33	1.12	3.04	2.65	2.97

Legend: TT - talking time, ROR - real OR recorded under the adaptive-heuristic-based commercial dialer (baseline), RTT - real TT (excluding script fill time after call), RAR - real AR, SOR - OR obtained from simulation, STT - TT from simulation (excluding script fill-in time after call), SAR - AR obtained from simulation, DOR = SOR - ROR, DTT = STT - RTT, DAR = SAR - RAR

The observed relationships are further illustrated in the plots presented in Fig. 4, which enable a visual assessment of changes in workload and service quality over time.

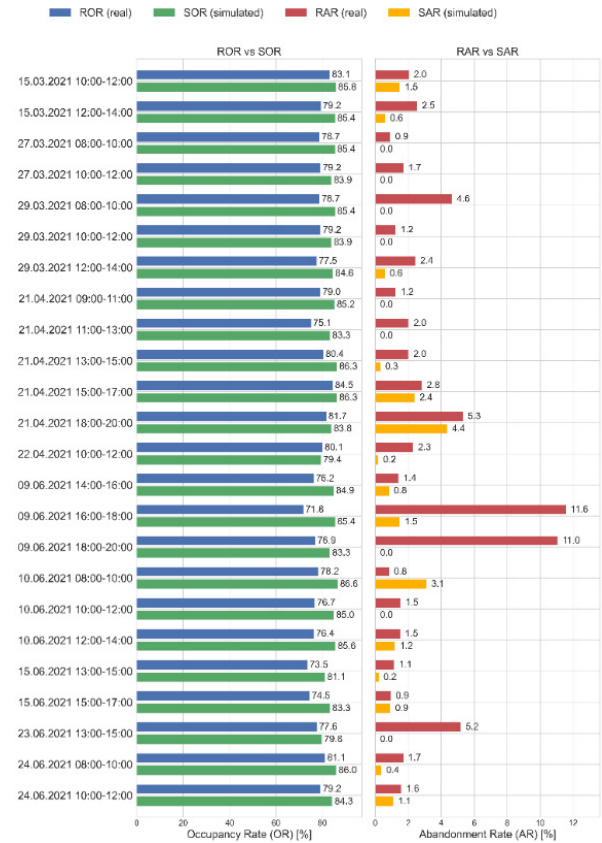


Fig 4. Comparison of Real and Simulated Performance Metrics Across Time Intervals

VII. CONCLUSIONS

The carried out study confirms the suitability of applying the TD3 algorithm for adaptive outbound traffic control in CC systems. In contrast to many approaches discussed in the literature review, which are based on analytical models, static rules, or short-term estimations, the proposed solution enables learning a control strategy directly from the observed system dynamics. The results indicate improved utilization of consultants' working time and an increase in the duration of actual customer interactions, while simultaneously maintaining a low and stable AR.

The proposed approach is scalable, universal, and independent of a specific system architecture. Therefore, in the authors' assessment, it can be regarded as an attractive solution for implementation in modern CC environments.

ACKNOWLEDGMENT

This work was carried out as part of the EMOTICA AI project, co-funded by the National Centre for Research and Development within the Smart Growth Operational Programme, Sub-measure 4.1.4 - Application Projects. Grant agreement No. POIR.04.01.04-00-0079/19.

In the process of preparing this manuscript, the authors made limited use of AI-based tools. M365 Copilot (GPT-5 chat model) was employed for editorial support. ChatGPT 5.3 was used solely for generating auxiliary Python code for bar chart visualization with Matplotlib. All scientific contributions, interpretations of results, and final decisions remain the sole responsibility of the authors

REFERENCES

- [1] M. Plaza, Ł. Pawlik, and R. S. Deniziak, "Call transcription methodology for contact center systems," *IEEE Access*, vol. 9, pp. 110975–110988, 2021, doi: 10.1109/ACCESS.2021.3102502.
- [2] Ł. Pawlik, M. Plaza, R. S. Deniziak, and E. Boksa, "A method for improving bot effectiveness by recognising implicit customer intent in contact centre conversations," *Speech Communication*, vol. 143, pp. 33–45, 2022.
- [3] M. Plaza, R. Kazała, Z. Koruba, M. Kozłowski, M. Lucińska, K. Sitek, and J. Spyryka, "Emotion recognition method for call/contact centre systems," *Applied Sciences*, vol. 12, no. 21, Art. no. 10951, 2022.
- [4] M. Ibrahim and N. B. Shroff, "Dynamic resource allocation for cloud-based call centers using reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1122–1135, 2019.
- [5] L. Feinberg, A. Mandelbaum, and B. Shalev-Shwartz, "Online learning for call center workforce management," *Operations Research*, vol. 66, no. 6, pp. 1674–1691, 2018.
- [6] S. Asghari, S. Yousefi, and M. S. Pishvae, "A data-driven optimization approach for call center performance improvement," *European Journal of Operational Research*, vol. 275, no. 2, pp. 676–690, 2019.
- [7] G. Koole, *Call Center Optimization*. MG Books, 2013.
- [8] N. Gans, G. Koole, and A. Mandelbaum, "Telephone call centers: Tutorial, review, and research prospects," *Manufacturing & Service Operations Management*, vol. 5, no. 2, pp. 79–141, 2003.
- [9] A. N. Avramidis and P. L'Ecuyer, "Modeling and simulation of call centers," in *Proc. Winter Simulation Conf.*, 2005, pp. 144–152.
- [10] W. M. N. Dilini, A. Karunaratne, and D. N. Ranasinghe, "Methods for controlling the call placement ratio for outbound dialing of a call center," in *Proc. Statistical Concepts and Methods for the Modern World Conf.*, Colombo, 2011, pp. 1–12.
- [11] N. Anisimov, N. Korolev, and H. Ristock, "Modeling and simulation of a pacing engine for proactive campaigns in contact center environment," in *Proc. Spring Simulation Multiconference*, 2008, pp. 249–255.
- [12] P. J. de Freitas Filho, G. F. da Cruz, R. Seara, and G. Steinmann, "Using simulation to predict market behavior for outbound call centers," in *Proc. Winter Simulation Conf.*, 2007, pp. 2247–2251.
- [13] S. Fourati and S. Tabbane, "Optimization of a predictive dialing algorithm," in *Proc. Advanced Int. Conf. Telecommunications*, 2010, pp. 212–218.
- [14] P. M. Amaral and M. M. Vital, "Predictive dialer intensity optimization using genetic algorithms," *Int. J. Machine Learning and Computing*, vol. 4, no. 3, pp. 286–291, 2014.
- [15] Aurus, "Predictive vs. progressive outbound dialers test results," 2023. [Online]. Available: <https://aurus5.com/blog/cisco/predictive-vs-progressive-outbound-dialers-test-results/> [Accessed: Jun. 20, 2026].
- [16] L. Qi, S. Ma, and K. Liu, "Research on predictive dialing system based on distributed call center," in *Proc. 4th Int. Conf. Software Engineering Research, Management and Applications (SERA'06)*, 2006, pp. 194–201, doi: 10.1109/SERA.2006.58.
- [17] V. B. Iversen, *Teletraffic Engineering Handbook*. Technical University of Denmark (DTU), 2015.
- [18] M. Zhang, Y. Sheng, N. Tian, W. Liu, H. Wang, L. Zhu, and Q. Xu, "Research and application of traffic forecasting in customer service center based on ARIMA model and LSTM neural network model," *J. Phys.: Conf. Ser.*, vol. 1881, Art. no. 032063, 2021.
- [19] M. Plaza and Ł. Pawlik, "Influence of the contact center systems development on key performance indicators," *IEEE Access*, vol. 9, pp. 44580–44591, 2021, doi: 10.1109/ACCESS.2021.3066801.
- [20] Ofcom, "Statement of policy on persistent misuse of electronic communications networks or services," 2006 (rev. 2016).
- [21] Federal Communications Commission, "47 C.F.R. § 64.1200: Delivery restrictions," *Electronic Code of Federal Regulations*, 2024.
- [22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.
- [23] C. Szepesvári, *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010.
- [24] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.
- [25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [26] M. Hu, "Problems with continuous action space," in *The Art of Reinforcement Learning*. Apress, 2023.
- [27] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," in *Proc. 29th Int. Conf. Machine Learning (ICML)*, 2012, pp. 179–186.
- [28] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learning Representations (ICLR)*, 2016. [Online]. Available: <https://arxiv.org/abs/1509.02971>
- [29] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Machine Learning (ICML)*, 2018, pp. 1587–1596.
- [30] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Machine Learning (ICML)*, 2018, pp. 1861–1870.
- [31] S. Liu, "An evaluation of DDPG, TD3, SAC, and PPO: Deep reinforcement learning algorithms for controlling continuous systems," in *Proc. 2023 Int. Conf. Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023)*, Atlantis Press, 2024.
- [32] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Machine Learning (ICML)*, 2014, pp. 387–395.
- [33] I. Smirnov and S. Gu, "RLBenchNet: The right network for the right reinforcement learning task," *arXiv preprint arXiv:2505.15040*, 2025.
- [34] Stable-Baselines3 Contributors, "Stable-Baselines3 documentation," 2021. [Online]. Available: <https://stable-baselines3.readthedocs.io> [Accessed: Jun. 20, 2026].
- [35] Z. Ben Hazem, F. Saidi, N. Guler, and A. H. Altaif, "A hybrid reinforcement learning framework combining TD3 and PID control for robust trajectory tracking of a 5-DOF robotic arm," *Automation*, vol. 6, no. 4, Art. no. 56, 2025.
- [36] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 32, no. 1, 2018.