Methodology

Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
1–21
© The Author(s) 2023
DOI: 10.1177/00375497231177123
journals.sagepub.com/home/sim

Knowledge-based and data-driven behavioral modeling techniques in engagement simulation

Zhi Zhu[1], Tao Wang[1], Hessam Sarjoughian[2], Weiping Wang[1] and Yuehua Zhao[3]

Abstract
As knowledge and data increase in scale and complexity, it is more difficult to apply these two key assets to achieve optimal effectiveness in engagement simulation. The aim of this study was to investigate the techniques of knowledge and data integration with respect to the development of smart agents to predict accurate behaviors in tactical engagements. To reduce the complexity of combat behavior representation, with respect to the functions, we represented subject matter expert operational knowledge by proposing multiple levels of cascaded hierarchical structure, namely, the function decision tree, to increase the readability and maintainability of the behavioral model. For decision points in a behavioral model, smart agents can be trained based on data samples collected from rounds of constructive simulations which provide validated physical models and tactical principles. As a proof of concept, we constructed a simulation testbed of multi-warhead ballistic missile penetration, which generated 129,600 constructive simulations over a total of 84 h. Thereafter, we selected 5817 data samples (i.e. ~4.5% of the simulations) using an operational metric of total rewards exceeding 100. The data samples are used to train an artificial neural network and then this network is used to develop a deep reinforcement learning agent. The results revealed that the training process iterated nearly 17,000 epochs until the policy loss decreased to an acceptable low value. The smart agent increased the ratio of ballistic missile target hits by 18.96%, a significant increase when compared with the traditional rule-based behavioral model.

Keywords
Agent-based systems, artificial intelligence, defense systems, deep reinforcement learning

1. Introduction

Over the past several decades, the capacity of artificial intelligence (AI) has steadily progressed to levels that match and outperform professional humans, as trained AI agents have succeeded in the games of chess, Go, and StarCraft II.[1,2] This trend is of significance to the military modeling and simulation (M&S) community, as games share numerous characteristics with combat simulations. Several commonly used combat simulations fall into three categories known as virtual, live, and constructive.[3] Constructive simulations are defined based on model granularity to campaign, mission, engagement, and engineering levels.[4] For the engagement simulation, models are tailored at a functional level of abstraction instead of using a set of hypothetical probabilities to represent input parameters in a mission-level simulation or processing signal-level information in an engineering-level simulation.[5]

In recent years, some researchers have used AI algorithms to develop engagement simulations to predict realistic behaviors. In June 2016, for example, an air combat agent referred to as Alpha AI defeated an experienced human pilot and demonstrated a significant advantage.[6] Similarly, in the final of the AlphaDogfight trials organized in August 2020, Heron systems outperformed seven competitors and won the championship; thereafter, it

[1]College of Systems Engineering, National University of Defense Technology, China
[2]School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, USA
[3]College of Information Management, Nanjing University, China

Corresponding author:
Tao Wang, College of Systems Engineering, National University of Defense Technology, Changsha 410073, Hunan, China.
Email: wangtao1976@nudt.edu.cn

readily defeated the fighter pilot in all five rounds of virtual battle.[7]

Modeling the dynamic behavior of the smart agent is among the challenges facing the development of engagement simulations. This includes the formulation of simulation and data modeling techniques requiring domain knowledge and observed data. Simulation modeling is a classical knowledge-based method for the development of causal relationships between a set of controlled factors and the corresponding set of responses.[8] This method is of significance and indispensable with respect to evaluating the effectiveness of combat engagements, given that actual battle scenarios are rarely observed in practice, as indicated by the low availability of historical data. Moreover, most of the military training data are classified or unobtainable due to unrealistic conditions such as hazardous test environments, high costs of trials, and limited availability of opposing platforms. It should be noted that simulation modeling may not have feasible physical principles or behavioral models under our current knowledge system.[9] It is also common knowledge, especially in the M&S community, that a simulation model is a certain aspect of representation for a given set of problems. Thus, it is generally influenced by various ideal assumptions or subjective factors and may omit critical factors when considering the ability and preference of different simulation modeling personnel.

As an alternative, establishing a data model based on observed data and optimizing the model through continuous iterations and training can gradually create a model that approaches the behavior of the actual system. Different from simulation modeling, the data model represents correlation relationships between a set of features and a set of tags. Although data modeling techniques can bypass domain knowledge to directly provide solutions to problems, they are not always capable of providing complete solutions due to their interpolative nature, and the validation and training data sets are typically randomly selected from the same data envelope. Furthermore, it can present challenges to developers, especially with respect to the design of systems with high safety and high accuracy and precision, as this method does not provide significant insight into physical systems.[10,11] Consequently, the data model cannot adapt to policy interventions and unexpected events, resulting in predictions that are generally inconsistent with some actual scenarios.

The main contribution of this study is the investigation of the application of the data-driven data modeling method to the prediction of smart behavior while providing insights into the knowledge-based simulation modeling method. A behavioral modeling approach referred to as the function decision tree (FDT) was designed and developed to represent behaviors in the form of a multi-cascaded hierarchical structure. Based on the behavioral model developed using FDT, smart agents can be trained using the data collected from constructive simulations of combat scenarios, thus allowing the agents to better select a policy to complete assigned missions. In addition, this study provides an experimental framework based on simulations with higher fidelity, which allows for fewer simulation runs and shorter time periods for model training via subject matter expert knowledge. Compared with most of the studies conducted by either simulation modeling or data modeling, the proposed knowledge-based and data-driven behavioral modeling method combines the knowledge and data domains. This provides a hybrid driving mode that benefits from both, which will play a significant role in future engagement simulations.

The remainder of this paper is organized as follows. Section 2 presents an investigation of the history of engagement simulation modeling development via a series of typical modeling paradigms, simulation platforms, and optimization algorithms. Section 3 details the proposed methodology with respect to the modeling architecture, FDT, and experimental framework. A case study of multi-warhead ballistic missile penetration is illustrated in Section 4, and the results and analysis are presented in Section 5, followed by the conclusions and scope of future research in Section 6.

2. Related work

In recent years, two well-known categories that are commonly accepted and widely used in the community are knowledge-based simulation modeling and data-driven data modeling, which led to various emerging methodologies and supporting technologies. These two modeling methods have been gradually combined to form a novel modeling paradigm, in addition to the rapid development of AI and big data, which is referred to in this paper as hybrid modeling using knowledge and data. A brief overview of these three categories of modeling techniques used in engagement simulation is shown in Figure 1.

2.1. Knowledge-based simulation modeling

With reference to the available literature, the knowledge-based simulation modeling method has undergone two main phases, that is, the standard specification-based phase and specific domain-oriented phase, as described below.

2.1.1. The standard specification-based phase. This phase is mainly focused on building standard specifications to technologically address the syntactical heterogeneity of different simulation modeling languages. The objective of building a standard specification is the development of a commonly accepted specification or formalism to promote structural model composability at the syntactical level with well-defined execution protocols (e.g. operational

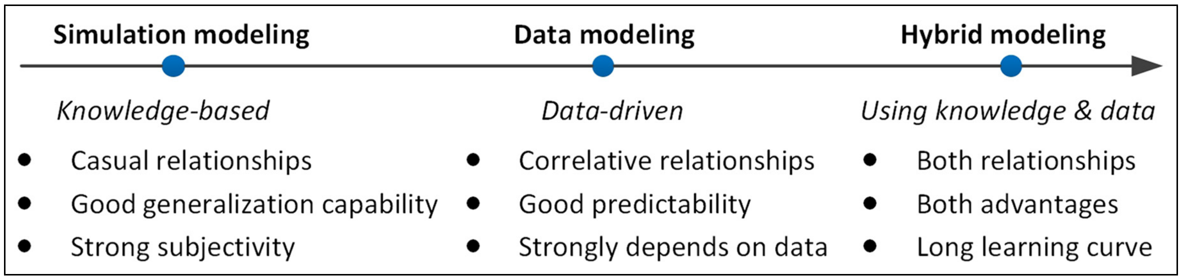

Figure 1. Three categories of modeling techniques used in engagement simulation.

semantics) but without domain-specific behavioral semantic. This phase comprises two main types of simulation modeling concepts:

- The first is with respect to the representation of the model static structure, which emphasizes the modularization and componentization of the model, with an aim to define open standard specifications in support of combining and reuse of existing simulation model libraries. Typically, there are common model specifications such as the High Level Architecture (HLA) proposed by Defense Modeling and Simulation Office in 1995,[12] Base Object Model (BOM) proposed by the Simulation Interoperability Standards Organization (SISO) in 1998,[13] and Simulation Model Portability proposed by European Space Agency in the mid-1990s, which was updated in 2011 and renamed the Simulation Modeling Platform 2.0 (SMP2).[14]
- The second is with respect to the model dynamic behavior, with a focus on the continuity between the conceptual model and the executable model in the life cycle of model development. In general, it applies formal and graphical modeling techniques to build behavioral models that can be simultaneously understood by humans and executed by computers. There are currently several formal approaches with respect to behavioral modeling, such as Petri nets,[15] finite-state machine,[16] and parallel functional decision tree.[17]

2.1.2. The specific domain-oriented phase. This phase specifies model elements and relationships based on semantics, which comprises two types of modeling concepts in support of model composability. They are the domain- and application-level model framework:

- The first involves the development of a domain level of the model framework representing the domain invariant knowledge (DIK) for a given system. Such widely used systems include the Extended Air Defense System,[18] System Effectiveness Analysis Simulation,[19] and Joint Theater Level Simulation.[20] In these systems, a model framework represents the model commonality and model generality at a high level of abstraction, prescribing the development of related subsystems or components.
- Second, based on a general simulation platform, an application level of the model framework representing Application Variant Knowledge is designed for a specific application problem. Such frameworks include the model framework of KD-HLA in the application of joint operation[21] and the model framework based on SMP2 in the sea warfare application.[22]

2.2. Data-driven data modeling

The above two mentioned phases of simulation modeling techniques are knowledge-based. However, with an increase in the capacity to collect, store, transmit, and process data in recent years, data have accumulated exponentially. Thus, it is necessary to investigate techniques for the effective analysis and mining of data. Unlike knowledge-based simulation modeling, such techniques are data-based and can be classified into two categories.

2.2.1. Data processing and analysis. The first is the traditional technology of data processing and analysis such as data farming and data mining. Data farming was proposed by the Marine Corps Combat Development Command in 1998[23] and has attracted significant attention from the worldwide military domain. In addition, multiple relative techniques have been updated and rapidly improved. It is the general consensus that data farming mainly includes five steps, namely, fertilization, cultivation, planting, harvesting, and regeneration.[24] Compared with data farming, data mining is a significantly more popular method of mass data processing with advanced theories and applications. It originated from the knowledge discovery in databases and was first reported at the 11th International Conference on Artificial Intelligence held in Detroit in August 1989. Data mining, which is referred to as knowledge discovery in databases, is a complex process of extracting and mining unknown patterns from a large amount of data.[25,26]

2.2.2. Machine learning algorithms. The second aspect includes a series of machine learning algorithms used for data analysis. Unlike traditional data modeling techniques, machine learning is concurrent with the rapid development of AI and big data, thus providing innovative means for data analysis. It mainly involves techniques for extracting patterns and models from structured data, that is, learning algorithms. In particular, numerous methods in data mining originate from machine learning, such as linear discriminative analysis,[27] decision tree,[28] artificial neural networks,[29] support vector machine,[30] deep learning,[31] and reinforcement learning,[32] among others.

In a survey of reinforcement learning, it models a given problem as the Markov process.[33] Based on dynamic programming and temporal-difference (TD) learning, many reinforcement learning algorithms have been developed, such as Deep Q-Network (DQN)[34] and Deep Deterministic Policy Gradient (DDPG).[35] DQN will inevitably lead to the overestimation of the real action value of Q network during training due to the limitations of TD learning. Hence, Double DQN[36] is proposed to alleviate the problem. Based on the Actor-Critic framework, DDPG performs not only well in continuous action space, but also faster than DQN in discrete action space. Many in-depth reinforcement learning algorithms with superior performance such as A3C (Asynchronous Advantage Actor-Critic),[37] TRPO (Trust Region Policy Optimization),[38] and SAC (Soft Actor-Critic)[39] have been proposed.

2.3. Hybrid modeling using knowledge and data

As previously alluded to, this type of modeling technique involves an attempt to build a smart agent based on knowledge and data. It is, therefore, critical to design a two-wheel hybrid-driven mechanism for the appropriate integration of knowledge and data. In particular, this concept emerged at the start of the big data era. For example, the theme of the Winter Simulation Conference (WSC) in 2014 was "Exploring big data through Simulation."[40] As in other scientific domains, modeling and simulation are anticipated to benefit from big data and deep learning.[41] Moreover, with reference to the available literature, similar hybrid-driven concepts include gray box modeling,[42] hybrid modeling,[43] complementary collaborative modeling,[44] and knowledge computing,[45] and data assimilation.[46,47] Regarding both knowledge and data as the basis of behavioral modeling, these modeling techniques make three significant contributions to the open literature.[48]

2.3.1. Incorporating data into knowledge-based modeling. Data facilitate knowledge-based modeling with respect to the reduction of input samples, estimation of unknown model parameters, and analysis of simulation results. The subjectivity of simulation models can be reduced to a significant extent while improving the accuracy of the system behavior prediction. This is highly suitable for systems that have accumulated rich domain knowledge over a long time period and are the preferred alternative for most advanced simulation systems. Typical studies include the hybrid application of simulation modeling and data analysis in the field of manufacturing and logistics,[49] and the integration of machine learning in the Discrete Event System Specification simulator to optimize the simulation execution time.[50]

2.3.2. Integrating knowledge into data-driven modeling. Knowledge assists data-driven modeling by providing simulation data sets. The main advantage is reducing the cost of data collection, and this method is appropriate for systems with a large amount of data, especially systems that require expensive or indirect data collection. Several published methods revealed that simulations are considered as the pre-processing stage of machine learning,[51,52] and simulations are used to generate a data stream for diagnosis analysis.[53]

2.3.3. Emphasizing the interoperation of knowledge and data. A common approach is referred to as digital twins.[54] It combines physical entities with their simulated (digital) counterparts. A digital model evolves in real time by continuously accepting data to achieve consistency with physical objects over the entire life cycle of model products. An example is a digital model for future aircraft proposed by the U.S. Air Force Laboratory.[55] Modeling and simulation techniques support the development of digital twins for analyses, predictions, diagnoses, and training important to the development, operation, optimization, and decision-making of their physical object counterparts.

2.4. Summary

In summary, based on these studies, it can be concluded that knowledge-based and data-driven modeling techniques serve as a technical foundation for the behavioral modeling of engagement simulations. However, it should be noted that the sole use of one or the other cannot accurately predict behaviors due to the increasing complexity of knowledge and data. Knowledge-based simulation modeling led to the development of numerous unified model specifications, modeling formalisms, and simulation protocols. In addition, multiple, extensively used composable modeling frameworks were established for various simulation systems. These methods solved the syntactical heterogeneity and semantic composability of models; however, there were several problems such as weak model

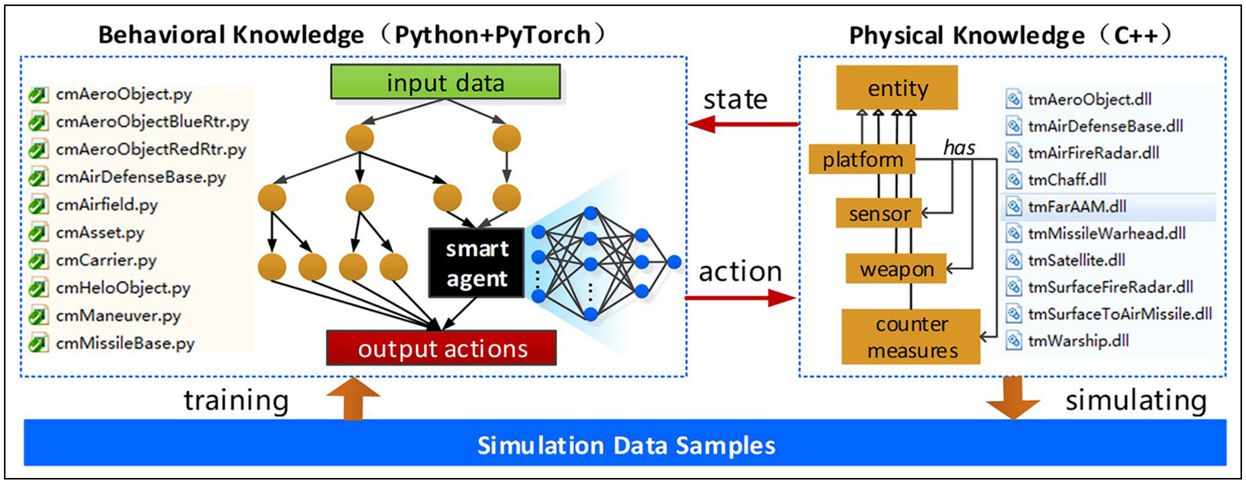

Figure 2. The knowledge-based and data-driven behavioral modeling architecture.

extensibility and maintainability, low degrees of model engineering, and a lack of intelligent modeling capability.

Data-driven data modeling allows for intelligent modeling using big data and AI. However, due to the complexity and unavailability of observed data in military simulations, such methods generally do not play an improved role. Hence, knowledge-based and data-driven behavioral modeling, as a combined means of using knowledge and data, is attracting significant attention across various research fields. This type of hybrid knowledge and data stands to provide superior advantages compared to the use of either knowledge or data. However, there are challenges associated with their combined use, and only a few systematic theoretical methods and technologies were developed for engagement simulations with reference to the available literature.

3. Methodology

This section presents a description of knowledge-based and data-driven behavioral modeling architecture in engagement simulations. Thereafter, the FDT method for behavioral modeling is detailed with respect to its definition, metamodel, and execution. For validation, an experimental framework is provided for comparative simulation experiments between the traditional experienced rule-based and the intelligent smart agent-based behavioral models.

3.1. Modeling architecture

The representation of a combat system is generally decomposed into physical knowledge and behavioral knowledge to reduce its analytic complexity.[56] Due to the different characteristics between these two types of knowledge, each involves unique modeling formalisms and programming languages. For example, physical knowledge generally involves the static structure and inherent dynamics of the system. In general, it is the most stable part and is readily manageable. At present, there are numerous modeling formalisms adopted to capture physical knowledge, such as class diagrams, static or dynamic data flow, discrete event, and differential equations, among others.

In contrast, behavioral knowledge is in the cognitive domain, where humans play an important role. Behaviors of systems are multifaceted and not simple to extract and share. Nevertheless, most modelers document it under tactical principles or if–then rules according to the decision-maker’s needs and preferences. These understandings are captured for use in inference engines. It should be noted that this study was focused on the representation of behavioral knowledge based on the premise that physical knowledge was appropriately implemented and was packaged as a collection of available dynamic link libraries.

Figure 2 presents the knowledge-based and data-driven behavioral modeling architecture in engagement simulation. As can be seen from this architecture, physical knowledge is represented as physical models that are finally implemented by C++, whereas behavioral knowledge is represented as cognitive models that are implemented in Python and PyTorch scripts.

On the right side of this figure, for example, the physical model is based on a set of basic modeling elements and relationships using Unified Modeling Language (UML), such as *platform*, *sensor*, *weapon*, and *countermeasures*. All of these elements are inherited by a common abstract element denoted as *entity*. Moreover, elements may be related, for example, a platform can be equipped with concrete sensors, weapons, and countermeasures.

On the left side, the behavioral model is mainly based on a tree structure and implemented into a list of scripts.

This tree structure receives data as inputs for internal computing in data nodes, utilizes a set of cascaded decision points to determine an output, and then translates the output to a concrete control action. In particular, a smart agent can provide appropriate control for each decision point, for example, an artificial neural network, a heuristic algorithm, or a fuzzy logic system.

Regarding interactions between the physical and behavioral blocks in this architecture, a set of pre-defined interfaces were developed to be called at appropriate instances. For example, the behavioral model uses the state information from the physical model and then provides the action instruction after a series of internal decision logic processes.

3.2. Function decision tree

Considering the complexity and uncertainty of behavioral modeling, it is difficult for traditional simulation modeling methods such as a rule-based system and state diagram to accurately reflect the actual combat behavior. As detailed in this section, the FDT was designed with respect to the function, including its basic definition, prescribed metamodel, and mechanism of execution. Using this novel method, simulation modelers are only required to consider the input and output of each decision node, and not the internal decision logic.

3.2.1. Definition. Different from decision tree in machine learning, the FDT organizes decision nodes according to the business process principle, whereas the decision tree considers attributes as nodes and divides them using a metric referred to as information gain. Moreover, decision tree is a type of machine learning algorithm that is generally used to solve classification problems, whereas FDT is focused on a concrete function that transforms inputs to outputs. The formal definition of FDT is presented below:

Definition 1: FDT is defined as a five-tuple, that is, $\mathrm{FDT} = (T, N_{data}, N_{action}, N_{decision}, \xi, \delta, \lambda)$, where T is the time set, N_{data} represents the set of data nodes, N_{action} represents the set of action nodes, $N_{decision}$ represents the set of decision nodes, $\xi : N_{data} \rightarrow N_{decision}$ represents the input function of data, $\delta : N_{decision} \times N_{data} \rightarrow N_{decision}$ represents the controlling flow or the decision-making function, and $\lambda : N_{decision} \times N_{data} \rightarrow N_{action}$ represents the action output function.

The design concept of FDT mainly considers three factors. First, most current simulation modeling methods consider the state and event as two critical elements, whereas FDT defines models with respect to the function. Hence, simulation modelers are not required to maintain a large number of states and events, among other variables, which can effectively alleviate the explosion problem of the state space.

Second, FDT adopts a hierarchical structure to describe the causal relationship and data flow between decision points. Thus, simulation modelers can promptly locate a node and change nodes, which leads to improved model maintainability and extensibility. Third, the nodes of FDT are abstracted at a higher level. In particular, the nodes can be considered black in capable of internal computing, as modelers do not require detailed knowledge of their operations.

3.2.2. Metamodeling. Based on the abovementioned FDT concept, a UML class diagram was developed to define its metamodel, as shown in Figure 3. In this diagram, the root denoted as *Tree* aggregates sets of *Node* and *Edge* elements. There are a total of three main types of tree nodes: *DataNode*, *DecisionNode*, and *ActionNode*. Note the *Tree*, *Node*, and *CompositeNode* are abstract and the latter two have a composition relationship. A generalization set {*complete, disjoint*} is used to constrain the relationships between the *Node* and the *ActionNode* and *CompositeNode*. For example, the node list is complete with only three types of nodes, so a {*complete*} constraint is added. Usually, there is no overlapping in these nodes. Therefore, the generalization set is also defined to have the {*disjoint*} constraint. For link relationships between nodes, a node has multiple incoming or outgoing edges, but an edge must have one source node and one target node.

3.2.3. Execution. After the design of the metamodel, the following step was to develop effective algorithms to ensure that the computation or execution of model elements is in accordance with the factual causal relationships in the physical world. In particular, the node causing the change of other nodes should be executed before the influenced node.

The FDT simulation execution algorithm is mainly divided into the decision logic control and data transmission processing parts (see Figure 4). The decision logic control is responsible for controlling the logic conforming to the causal relationship defined by the real world. For instance, as shown in Figure 4(a), an available decision logic is R to D2, D2 to D3, and D3 to A4 in the order given, where R represents the root node, D represents a decision node, and A represents an action node. The data transmission processing is responsible for sorting the data nodes to ensure that the data nodes as input are calculated before the data nodes as output. For instance, an available topological sorting should be A or B or C or G is calculated first, then D or E is calculated, F is calculated, and finally H is calculated as shown in Figure 4(b).

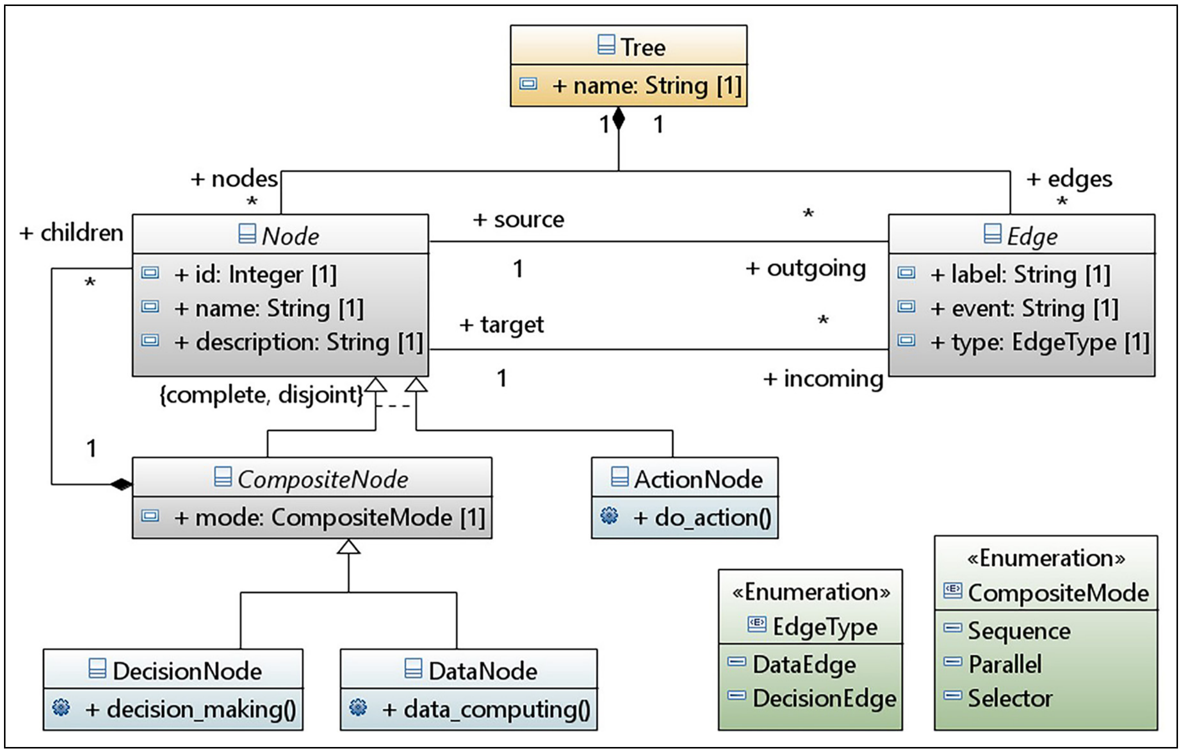

Figure 3. The FDT metamodel.

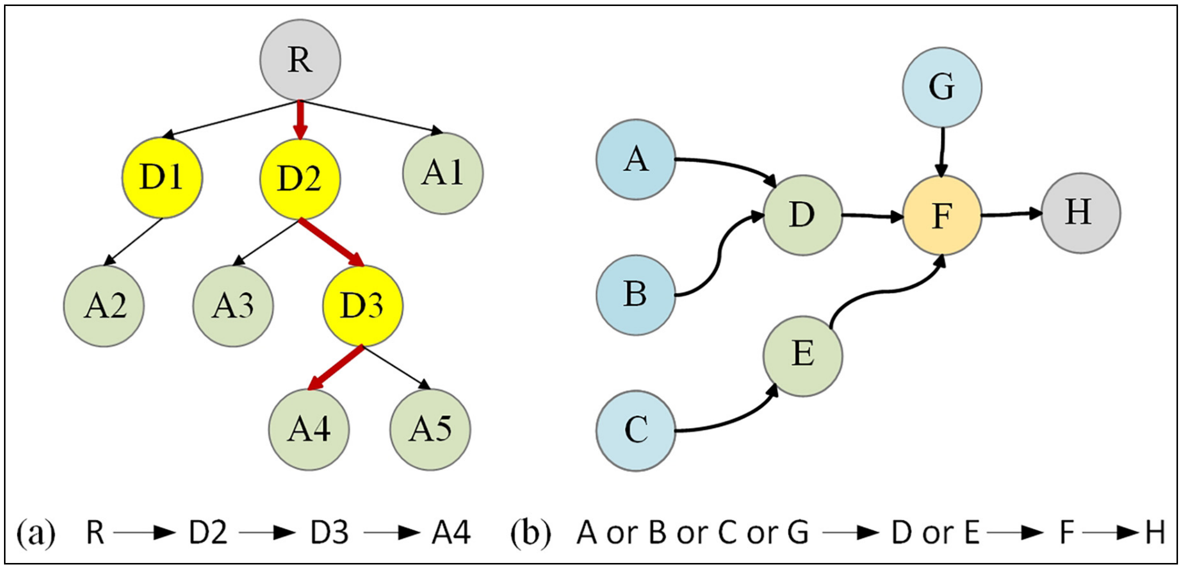

Figure 4. Two instances of the FDT execution for (a) decision logic control and (b) data transmission processing.

The decision logic control is shown in Algorithm 1. The initialization is responsible for the construction of FDT, which is mainly used to construct the parent–child relationships among nodes. The decision node accepts the data output from data nodes and then generates the subsequent decision or action node to be executed based on the internal decision logic. The action node directly executes concrete actions.

A data node is mainly responsible for computing data and transmitting the data to other nodes. In this process, the input–output relationship of data nodes is in the form of forward computing, to ensure accurate causal

Algorithm 1. Decision logic control.

```
# Initialization of the FDT
define a node with its list of inputs
  **for** node in inputs **do**
      append self into the outputs of node
  **end**
  # Execution of the FDT
  get the root of FDT
  **if** root is a decision node **then**
      **if** conditions are satisfied **then**
          iterate and scan the sub-tree
      **else** return
  **else if** an action node **then**
      do corresponding actions
  **else** throw an error of "*unknown type of node*"
```

FDT: function decision tree.

Algorithm 2. Data transmission processing.

```
# Initialization of the list of sorted nodes
define a tree as a given FDT dictionary
**while** tree is not none **do**
    obtain all nodes with outputs $o$
    obtain all nodes with inputs $i$
    list $A = o - i$ # nodes that have only outputs but with no inputs
    list $B = i - o$ # nodes that have only inputs but with no outputs
    **if** list $A$ is not empty **then**
        append a random node in $A$ into list $S$
        delete the random node from the FDT dictionary
        **for** node in $A$ output nodes **do**
            **if** node is in list $B$ **and not** in list $S$ **then**
                append it to list $S$
    **else** throw an error "*FDT contains circular structure*"
    **return** list $S$
**end while**
```

FDT: function decision tree.

relationships. Hence, all nodes should be topologically sorted, as expressed by Algorithm 2; thus, the final result can be obtained by scanning each node in the sorted nodes.

3.3. Experimental framework

After the appropriate preparation of the physical and behavioral models, experiment design can be conducted to generate data samples based on Monte Carlo simulation runs. The data samples are then used to train a smart agent based on a specific intelligent algorithm, and this trained smart agent can be called in the same simulation scenario, thus forming a closed loop of the experimental framework. As shown in Figure 5, the experimental framework of intelligent training and testing mainly includes four workflows.

First, the workflow of the initial experiment demonstrates the development of a simulation application based on several general steps, each of which has a corresponding tool. For example, we used a tool referred to as *DataManager* to manage a large amount of prototype data, adopted *MagicDraw* to design a conceptual behavioral model, and applied *PyCharm* to implement the conceptual model into executable codes. The prototype data and behavioral model are integrated into the *ScnEditor* to automatically instantiate and compose relevant model components. When a scenario is developed, an experiment can be designed in the *ExperimentManager*. This step involves the selection of factors and responses for the generation of experimental alternatives by a specific experiment design method such as comprehensive design, orthogonal design, and Latin hypercube design. Thereafter, Monte Carlo simulation runs can be executed to generate data for statistical analysis and visual display using the *SimAnalysis* and *SimDisplay* tools, respectively.

Second, the data generated from the previous step are collected and pre-processed in the workflow of machine learning (ML) with the objective of training a smart agent. In general, the generated simulation data typically contain invalid data, despite the employment of an appropriate experiment design method that allows for fewer simulation runs. Hence, it is necessary to remove invalid data items or translate the cyclical features into sine and cosine components to increase model performance and allow for the faster training of agents. Before training, data samples are classified into training, validation, and test sets, as follows. In particular, a larger part is allocated for training and less for validation using a cross-validation technique. The remaining data are then used for testing. Once the loss function converges to an accepted value, we terminate the training process, and the trained model is saved for the following deep reinforcement learning (DRL) workflow.

Third, the DRL process loads the initial policy network trained from ML and continuously interacts with the combat environment to learn an improved policy. A policy represents the basis upon which an agent selects an action in a given state. Based on this policy, an agent can produce an action to change the outside environment and simultaneously receive a reward, thus constructing a connection between the state, action, and reward as a form of memory for future decision reference.

Forth, several comparative experiments were designed to evaluate the performance of the trained agent by constructing a basic scenario for the same initial setting. The only difference was the decision models to be called. The first was the best rule-based decision model analyzed from the simulation results of the initial experiment. The second was the smart agent learned from the filtered data samples

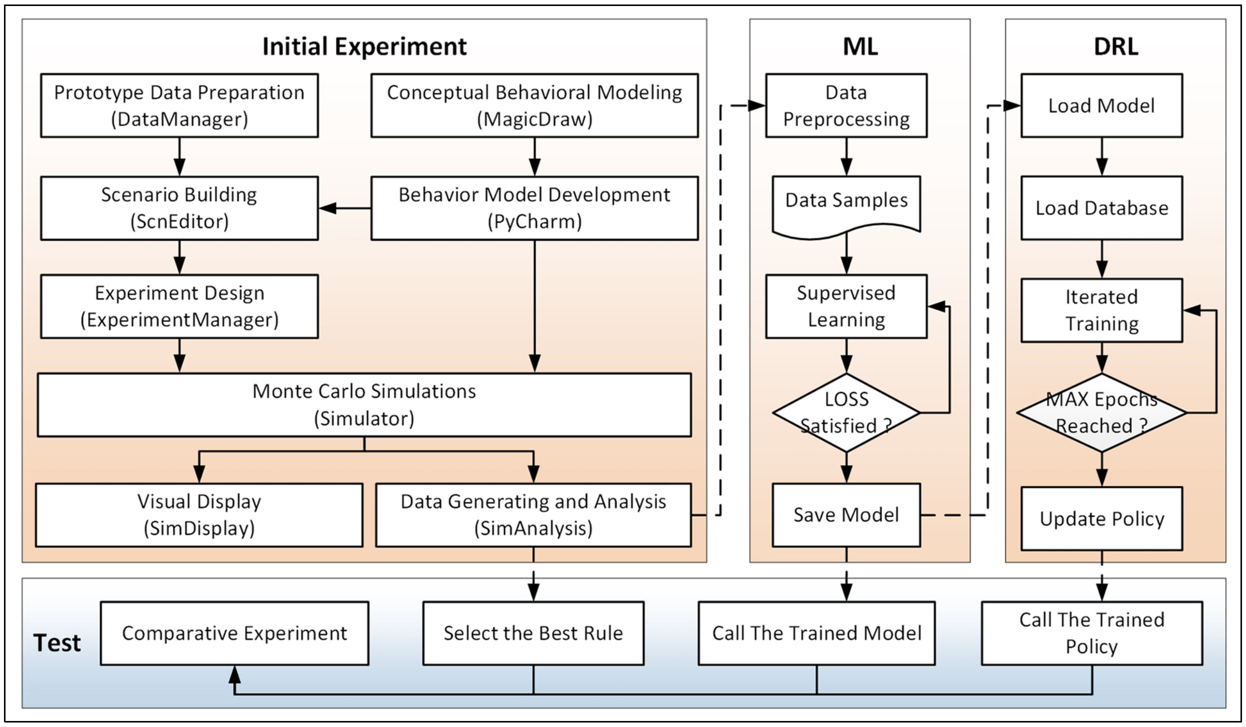
Initial Experiment
Prototype Data Preparation (DataManager)
Conceptual Behavioral Modeling (MagicDraw)
Scenario Building (ScnEditor)
Behavior Model Development (PyCharm)
Experiment Design (ExperimentManager)
Monte Carlo Simulations (Simulator)
Visual Display (SimDisplay)
Data Generating and Analysis (SimAnalysis)
ML
Data Preprocessing
Data Samples
Supervised Learning
LOSS Satisfied ?
Save Model
DRL
Load Model
Load Database
Iterated Training
MAX Epochs Reached ?
Update Policy
Test
Comparative Experiment
Select the Best Rule
Call The Trained Model
Call The Trained Policy

Figure 5. The experimental framework of intelligent training and testing.

Table 1. Force composition of both sides.

Side	Entity	Number	Description
RED	Launching silo	1	A launching base that contains ballistic missiles
	Ballistic missile	1	A booster rocket that contains six warheads
BLUE	C2 center	1	A center of command and control for battle management
	Forward radar	2	Two forward radars that provide early warning information
	GBI base	1	An intercepting base equipped with 44 air defense missiles
	Warship	1	A warship equipped with 36 surface-to-air missiles
	Satellite	3	Three geostationary earth orbit satellites
	Cities	6	Six important targets with high value

GBI: ground-based interceptor.

using the ML method, and the third was the trained policy based on the DRL.

4. Case study

This section details the combat scenario, presents the ballistic missile model, and specifies the smart DRL agent used for the intelligent target assignment problem.

4.1. Scenario description

To develop a simulation application, it is necessary to apply a simulation system with a high fidelity and extensible model. In this study, we applied Weapon Effectiveness Simulation System (WESS)[56] to conduct simulations and demonstrate the integration of smart agents.

4.1.1. Combat configuration. In the simulation scenario, the red side considers six important blue side cities as its targets to destroy by launching a ballistic missile with six intelligent warheads. The goal of the blue side is to prevent ballistic missile attacks on the cities. Hence, the blue defense systems are required to identify and locate the incoming enemy missiles within a minimal time period and realize early interception. The forces on both sides are listed in Table 1. The blue side has a relatively complete defense system composed of multiple cross-domain

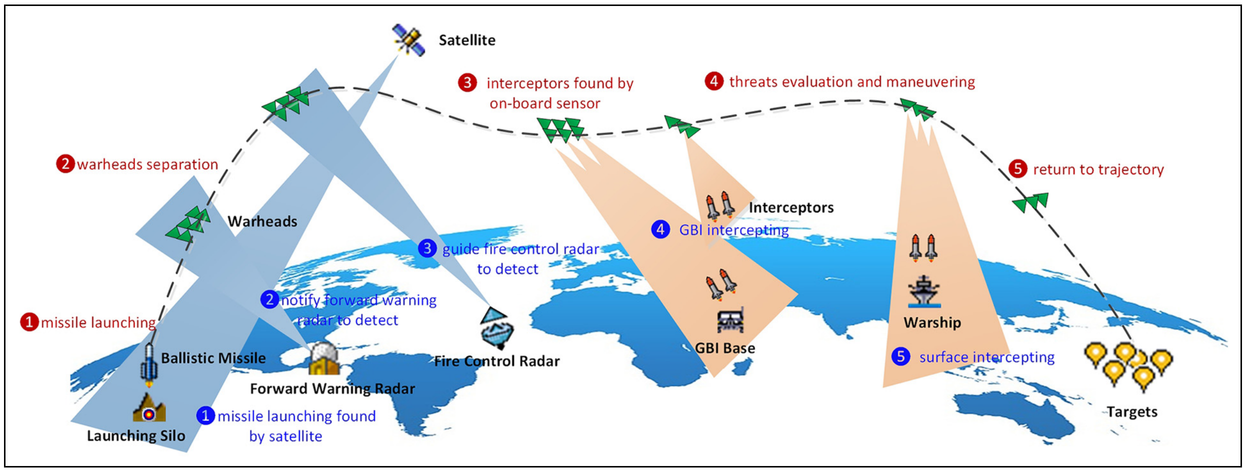

Figure 6. The combat process of ballistic missile penetration.

defense devices such as satellites, radars, warships, ground-based interceptor (GBI) base, and center of command and control (C2).

Figure 6 presents the combat process of the simulation scenario. As can be seen from this figure, the scenario considers several classical tactics of attack and defense engagement. For example, the red side mainly includes cooperative detection, threat assessment, orbit maneuver, target assignment, and return to orbit. The blue side mainly includes defense devices composed of a System of Systems with the objective of protecting assets from incoming threats.

At the start, the launching silo launches a ballistic missile. After the booster stage, the loaded warheads are released. Each warhead has a sensor, which is turned on when approaching some pre-defined location. Based on the detected information, the warhead may maneuver to change its orbit.

In the booster stage of the ballistic missile, the early warning satellite captures the infrared radiation and reports to the C2 center of the defense side. After acquiring the flight trajectory information of the incoming missile, the C2 center transmits it to the early warning radar for further detection and tracking. Using the data from early warning radars, the fire and control radar maintains tracking using more accurate data. When the predicted interception point accuracy exceeds the pre-determined threshold, the air defense base launches missiles to intercept the incoming threats.

After launching interceptors from blue-side defense systems, on-board sensors from red-side warheads may identify them via intelligent cooperative detection. Thereafter, the warhead performs threat assessments according to its own state and the detected interceptor status. When the most threatening target is identified, the warhead makes decisions with respect to several aspects such as the timing, duration, and direction of maneuverability. If the penetration succeeds, it will re-allocate the targets for those successful warheads, return along the original orbit, and finally destroy the designated targets.

4.1.2. Physical computation. As this is within the scope of the physical domain, and not considered as the focus of this study, a brief overview of the physical aspects of the ballistic missile model is presented. In the launching coordinate system, a three-degree-of-freedom ballistic missile model is expressed by equation 1

$$\begin{cases} mv' = R + G + P + F_{ECI} + F_{KI} + F_m \\ x' = v \end{cases} \quad (1)$$

where v and x represent the vectors of the velocity and position, m represents the mass, P represents the thrust, R represents the aerodynamic force, G represents the gravity, F_{ECI} represents the centrifugal inertial force, F_{KI} represents the Coriolis force, and F_{m} represents the control force related to the current state of the missile.

The aerodynamic force $R_v(-D, L, 0)$ in the velocity coordinate system requires translation to R in the launching coordinate system

$$\begin{cases} Q = 0.5 \times \rho \times V^2 \\ D = C_D \times Q \times A_{rREF} \\ L = C_L \times \alpha \times Q \times A_{rREF} \end{cases} \quad (2)$$

where ρ represents the air density, V represents the speed, C_D represents the drag coefficient, C_L represents the lift coefficient, α represents the attack angle, and A_{rREF} represents the reference area.

The gravity $G(0, -G, 0)$ in the NUE (North, Up, East) coordinate system requires translation to G in the launching coordinate system

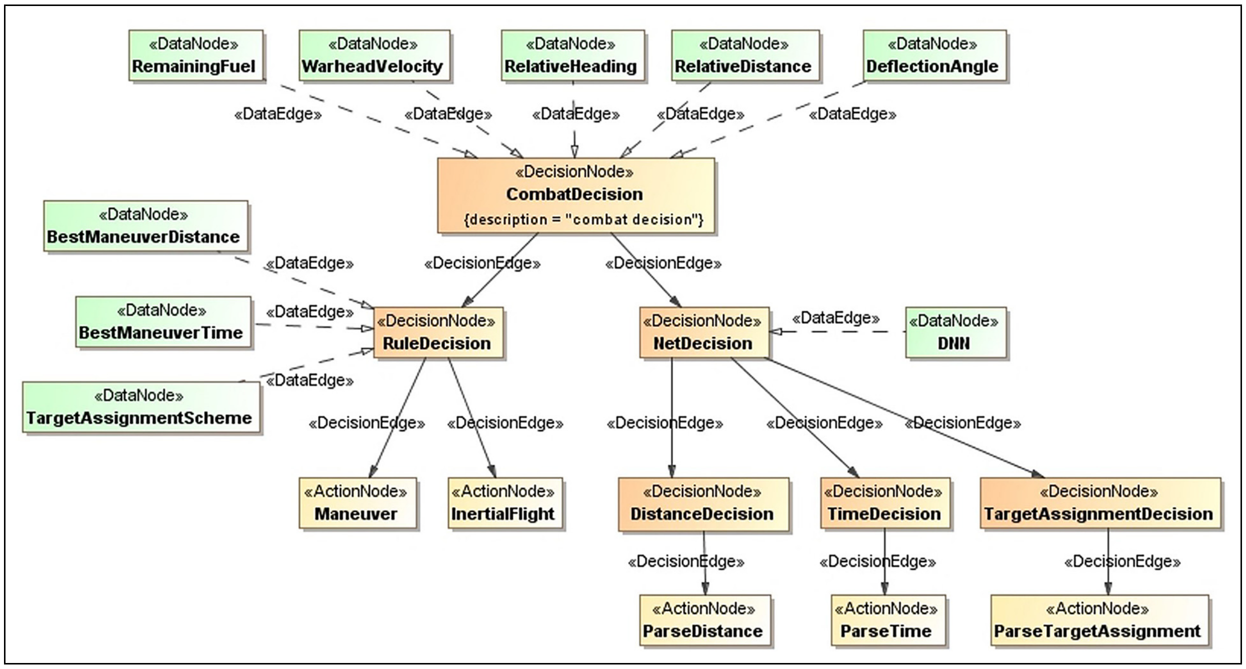
«DataNode» RemainingFuel
«DataNode» WarheadVelocity
«DataNode» RelativeHeading
«DataNode» RelativeDistance
«DataNode» DeflectionAngle
«DataEdge»
«DecisionNode» CombatDecision {description = "combat decision"}
«DataNode» BestManeuverDistance
«DataNode» BestManeuverTime
«DataNode» TargetAssignmentScheme
«DecisionEdge»
«DecisionNode» RuleDecision
«DecisionNode» NetDecision
«DataNode» DNN
«ActionNode» Maneuver
«ActionNode» InertialFlight
«DecisionNode» DistanceDecision
«DecisionNode» TimeDecision
«DecisionNode» TargetAssignmentDecision
«ActionNode» ParseDistance
«ActionNode» ParseTime
«ActionNode» ParseTargetAssignment

Figure 7. The behavioral model of the ballistic missile developed using the FDT method.

$$G = \frac{mG_C}{(RE + h)^2} \tag{3}$$

where $G_C = 3.986 \times 10^{14}$ ($\mathrm{m}^3/\mathrm{s}^2$), h (m) is the height, and RE (m) is the earth radius.

The thrust $P(p_h, 0, 0)$ in the body coordinate system requires translation to P in the launching coordinate system

$$P_n = \begin{cases} P_0 + S_n(p_0 - p_h) & n = 1 \\ P_0 - S_n p_h & n = 2, 3 \end{cases} \tag{4}$$

where P_0 represents the pure thrust, P_n represents the n-stage vacuum thrust, S_n represents the n-stage nozzle area, p_0 represents the surface standard atmospheric pressure, and p_h represents the h-height atmospheric pressure.

The centrifugal inertial force F_{ECI} in the Earth-Centered, Earth Fixed (ECEF) coordinate system requires translation to the launching coordinate system

$$\begin{cases} F'_{ECI} = -mW'_{EE} \\ W'_{EE} = \omega \times (\omega \times v) = (\omega \cdot r)\omega - \omega^2 r \end{cases} \tag{5}$$

where r is the geocentric vector diameter of missile, and ω is the rotational acceleration of the earth.

The Coriolis force F_{KI} in the ECEF coordinate system requires translation to the launching coordinate system

$$\begin{cases} F'_{KI} = -mW'_{KE} \\ W'_{KE} = 2\omega v \end{cases} \tag{6}$$

where ω is the rotational acceleration of the earth, and v represents the velocity of the missile.

4.1.3. Behavioral representation. Based on the proposed FDT method, the behavioral model of the ballistic missile was developed using the mechanism of the UML profile, as shown in Figure 7.

In particular, there are three types of stereotypes with respect to three key modeling elements of the FDT meta-model (see Figure 3), including $\ll$*DataNode*$\gg$, $\ll$*DecisionNode*$\gg$, and $\ll$*ActionNode*$\gg$. Moreover, this profile defines two types of stereotypes, that is, $\ll$*DataEdge*$\gg$ and $\ll$DecisionEdge$\gg$, which represent data transmission and decision control, respectively. Concretely, data transmission is represented by a dashed line with a hollow triangle as an arrowhead, and decision control is represented by a solid line with a solid arrowhead.

The root of the behavioral model is the *CombatDecision* node, which applies the $\ll$*DecisionNode*$\gg$stereotype. This node is used to determine the decision path that the model selects based on the current battle scenario, in addition to the performance of the ballistic missile. Hence, it receives five data nodes as its inputs, namely, *RemainingFuel*, *WarheadVelocity*, *RelativeHeading*, *RelativeDistance*, and *DeflectionAngle*, and two decision nodes as its outputs, that is, *RuleDecision* and *NetDecision*. If this model selects the rule decision process, it uses a priori knowledge as its inputs, such as the optimal distance and optimal time to initiate the maneuvering action, in addition to the predefined scheme of target assignment. Thereafter, this model performs definite actions such as automatic maneuvering or inertial flight. In contrast, if the net decision is selected, the model integrates a deep neural network (DNN), which is a data node. Based on the data node, the model further determines an improved distance and time of maneuvering, or an improved scheme for target assignment. Furthermore, this model parses the net outputs and then performs relevant actions.

4.2. Smart agent

4.2.1. Target assignment. In multi-warhead ballistic missile penetration, if a ballistic missile carries m warheads fighting against n targets, the target assignment matrix is $X = [x_{ij}]$, where $0 \leqslant i < m$, $0 \leqslant j < n$, and $x_{ij} = 0$ or 1. When $x_{ij} = 1$, this indicates that Target t_j is assigned to Warhead w_i; otherwise, it is unassigned. In the phase of target assignment, each target should be assigned a minimum of one warhead, and each warhead should have one target to attack; thus, $\sum_{j=0}^{n-1} x_{ij} \geqslant 1$ and $\sum_{i=0}^{m-1} x_{ij} = 1$. If r_{ij} is set as the reward for an attack of Warhead w_i on Target t_j, and the maximization of total rewards is considered as the assignment goal, the target assignment model is established as follows

$$\max \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} r_{ij} \cdot x_{ij} \qquad s.t. \begin{cases} \sum_{i=0}^{m-1} x_{ij} = 1 \\ \sum_{j=0}^{n-1} x_{ij} \geqslant 1 \\ x_{ij} \in \{0, 1\} \end{cases} \tag{7}$$

Consider the reward r_{ij} where target t_j is assigned to warhead w_i. Its design should be in accordance with the following principle: different schemes of target assignment have a significant influence on the final result of engagement. That is, the result of target assignment should be sensitive to the RTH (Ratio of Target Hits) index and an executable goal function that is translated from the commander's intent.[57,58]

Before the target assignment, whether to perform successful penetrations for a ballistic missile when facing rounds of interceptions is another factor that should be considered. Therefore, the final reward is composed of two phases of rewards, that is, the penetration phase and the mission completion phase.

In the penetration phase, there are two scenarios, as expressed by equation 8. If a warhead penetrates successfully ($p = 1$), the reward is dependent on the consumed energy, the number of successful penetrations, and the miss distance. Otherwise ($p = 0$), the reward is directly assigned a relatively large penalty of -50 based on the operational knowledge of subject matter experts

$$R_p = \begin{cases} r_{energy} + r_{stage} + k_n \cdot \frac{r_{iKillRadius}}{d_{iMissDist}} & p = 1 \\ -50 & p = 0 \end{cases} \tag{8}$$

where $r_{energy} = k_m \cdot (m_{\max} - m)/m_{\max}$ represents the reward of consumed energy, where k_m is the relative coefficient of energy consumption; $m_{\max}$ represents the maximum amount of carrying energy; m represents the available energy; r_{stage} represents the number of successful penetration events; k_n represents the relative coefficient of miss distance; $r_{iKillRadius}$ represents the kill radius; and $d_{iMissDist}$ represents the miss distance.

At the phase of mission completion, two scenarios should be considered. The first is that wherein a warhead hits the target successfully, and the second is that wherein a warhead misses the target. Similar to the variable p defined at the phase of penetration, we set $m = 1$ to represent the first scenario, and $m = 0$ to represent the second scenario. This phase reward is then computed as follows

$$R_m = \begin{cases} 5 + 5 \cdot \frac{r_{wKillRadius}}{d_{wMissDist}} & m = 0 \\ 40 + 10 \cdot k_{imp}\left(1 - \frac{d_{wMissDist}}{r_{wKillRadius}}\right) & m = 1 \end{cases} \tag{9}$$

where $r_{wKillRadius}$ represents the kill radius of the warhead, $d_{wMissDist}$ represents the miss distance of the warhead, and k_{imp} is a coefficient that represents the importance of different targets within a range of [0, 1].

Note that this coefficient and constants -50, 40, 10, and 5 are determined based on subject matter experts input to improve the influence of target assignment on the final RTH index.

4.2.2. DNN architecture. Consider the state space, a set of input parameters used for the neural network, consisting of the remaining fuel of the warhead, the relative distance between the warhead and target, the velocity of the warhead, the relative heading angle, and the deflection angle between the warhead and target.

These parameters are selected based on subject matter expert operational knowledge. The remaining fuel is directly related to the energy available to the ballistic missile. The relative distance determines whether the warhead has sufficient energy to maneuver. In particular, the velocity of warhead can increase or decrease the warhead effectiveness of penetration, depending on the relative heading angle. However, the relative heading angle cannot provide a complete account with respect to positioning, as it is dependent on the deflection angle to determine whether the warhead is approaching or far away from the target. It is noteworthy that the RTH index is defined as a probability of success, ranging from 0% to 100%, for the success of the ballistic missile on penetration missions and

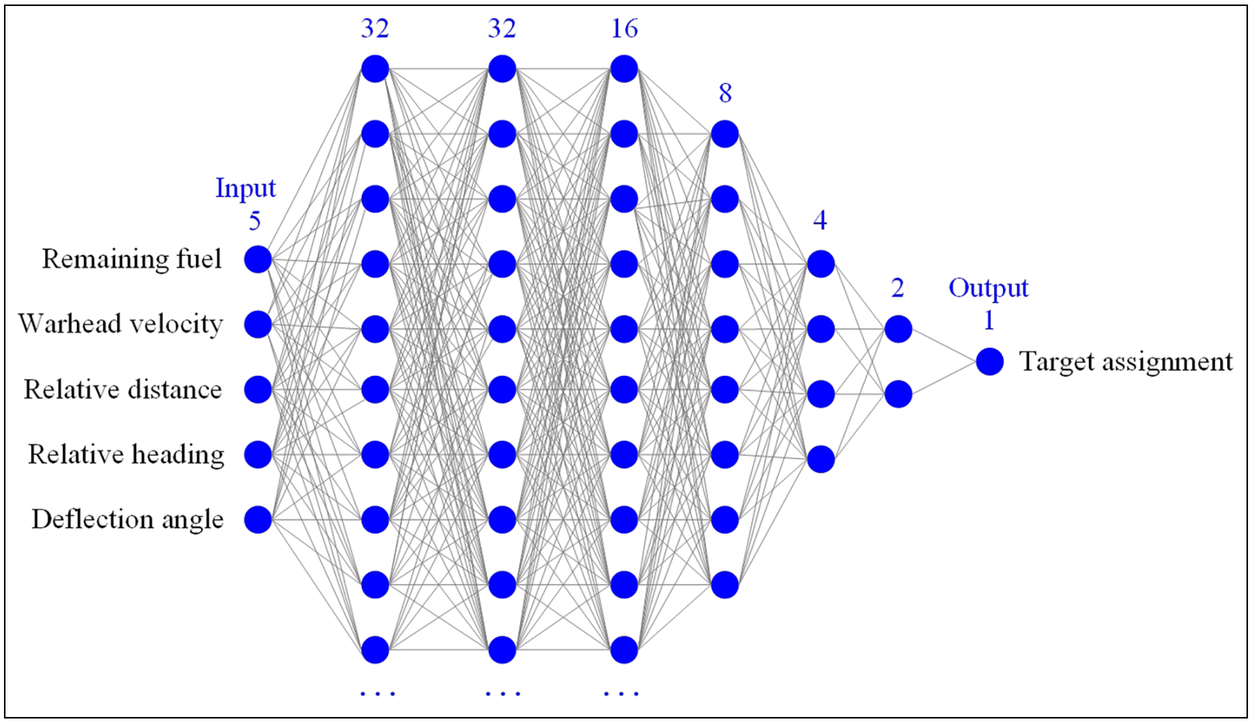

Figure 8. The DNN architecture.

Table 2. The parameters considered in the state space.

Name	Variable	Min	Max	Unit	Scaling
Remaining fuel	f	0	60	kg	f/60
Warhead velocity	v	500	1000	m/s	(v – 500)/500
Relative distance	r	0	1500	km	r/1500
Heading angle	phi	− 180	+180	°	sin(phi)
Deflection angle	theta	− 180	+180	°	cos(theta)

destroying as many targets as possible. This is implemented by maneuvering to defend the warheads from opposing interception.

In addition, the DNN model is formed by eight layers of nodes (six hidden layers), and its structure is shown in Figure 8. All of the nodes have a rectified linear activation, for example, the ReLU[59] function.

To increase the model performance, feature engineering techniques are employed to pre-process these parameters before model training, as detailed in Table 2. A common technique is feature scaling, which is performed to equally distribute the importance of each feature in the DNN learning process. This is realized via the scaling and translating of all features within the range of 0–1, thus decreasing the influence due to different units and different scales between different features. In addition, to better handle cyclical features, the angles related to heading and deflection are directly encoded into their sine and cosine components.

For the action space, we consider a ballistic missile carrying six warheads to six targets. Hence, there are a total of 720 schemes for target assignment, and each scheme is encoded into a number ranging from 1 to 720 in sequence, as detailed in Table 3. For example, No. 1 represents the assignment scheme "123456," which is decoded into the following commands when the trained model is called: "Target A is assigned to Warhead 1, Target B is assigned to Warhead 2, etc."

In the DNN architecture, the network outputs an available prediction of target assignments, tags are the factual assignment schemes of simulation data samples, and the supervision signal is the cross entropy of probability distributions between the predicted and the sampled schemes, as shown in Figure 9.

Table 3. The parameters considered in the action space.

Name	Variable	Min	Max	Unit	Parsing
Assignment scheme	s	1	720	N/A	A_6^6 = 720 schemes

N/A: not applicable.

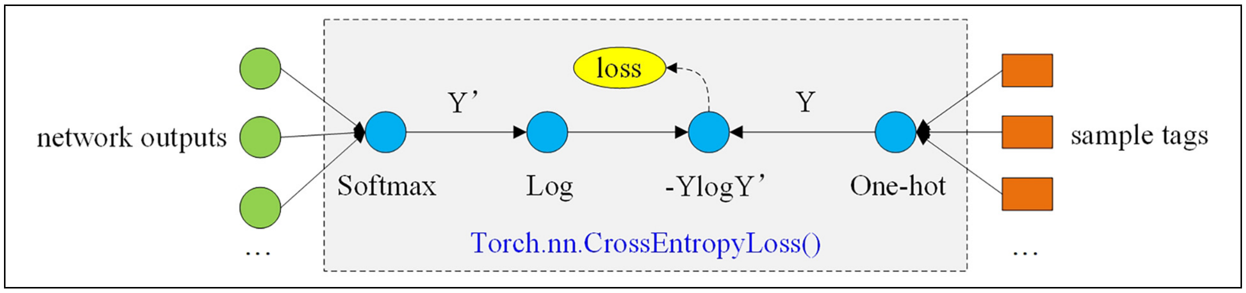

Figure 9. The supervised learning framework of multiple classification.

Different from binary classification, the supervised learning framework for multiple classification is used. In binary classification, the loss function is computed as follows

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)] \qquad (10)$$

where N is the number of data samples, y_i is the tag value of the ith data item, and p_i is the predicted value of the ith data item through a run of feedforward propagation. In particular, it is the result of a network activated by the sigmoid function.

In the multi-classification problem, the loss function is computed as follows

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K} y_{ic}\log(p_{ic}) \qquad (11)$$

where N is the number of data samples, K is the number of classifications, and p_{ic} is the probability that the ith data item belongs to the cth classification, which is subjected to $\sum_{c=1}^{K} p_{ic} = 1, i = 1, 2, ..., N$.

Unlike binary classification, it is the result of that network that is performed by the Softmax function. In addition, y_{ic} is the result of the tag value performed by one-hot encoding.[60] If the tag value of the ith data item is equal to the cth classification, the corresponding position is 1; otherwise, it is 0.

4.2.3. DDPG algorithm. Deep Deterministic Policy Gradient is a model-free off-policy algorithm based on DNNs and actor-critic policy, as shown in Figure 10. As it uses a policy network to generate a deterministic action instead of sampling based on the probability distribution of actions, it is a type of deterministic policy. Similar to the actor-critic method, it involves two networks. The first is the actor, which proposes an action given a state. The second is the critic, which predicts if the action is acceptable or unacceptable.

To improve the training stability, it uses two target networks to slowly update the policy. It finds the action that maximizes $Q(s_t, a_t)$ before back propagation and does not use its maximum value directly, thus rendering the estimated targets more stable. In addition, it uses experience replay by storing a list of tuples (state, action, reward, and next state). Instead of learning only from recent experience, the model learns from sampling all the accumulated experience. To implement the improved exploration by the actor network, noisy perturbations, that is, $a_t + n$, are used, especially an Ornstein–Uhlenbeck process for generating noise.[61]

On one hand, for the update mechanism, this algorithm defines the critic loss based on the cross entropy, as previously mentioned, where $r_{t+1} + Q(s_{t+1}, a_{t+1})$ is the expected return obtained by the target network, and $Q(s_t, a_t)$ is the action value predicted by the critic network. On the other hand, it defines the actor loss as computed using the value provided by the critic network for the actions performed by the actor network, which maximizes this quantity.

After training is complete, the model is selected as the initial policy of the next reinforcement learning. When calculating the state value function, a measure of entropy is

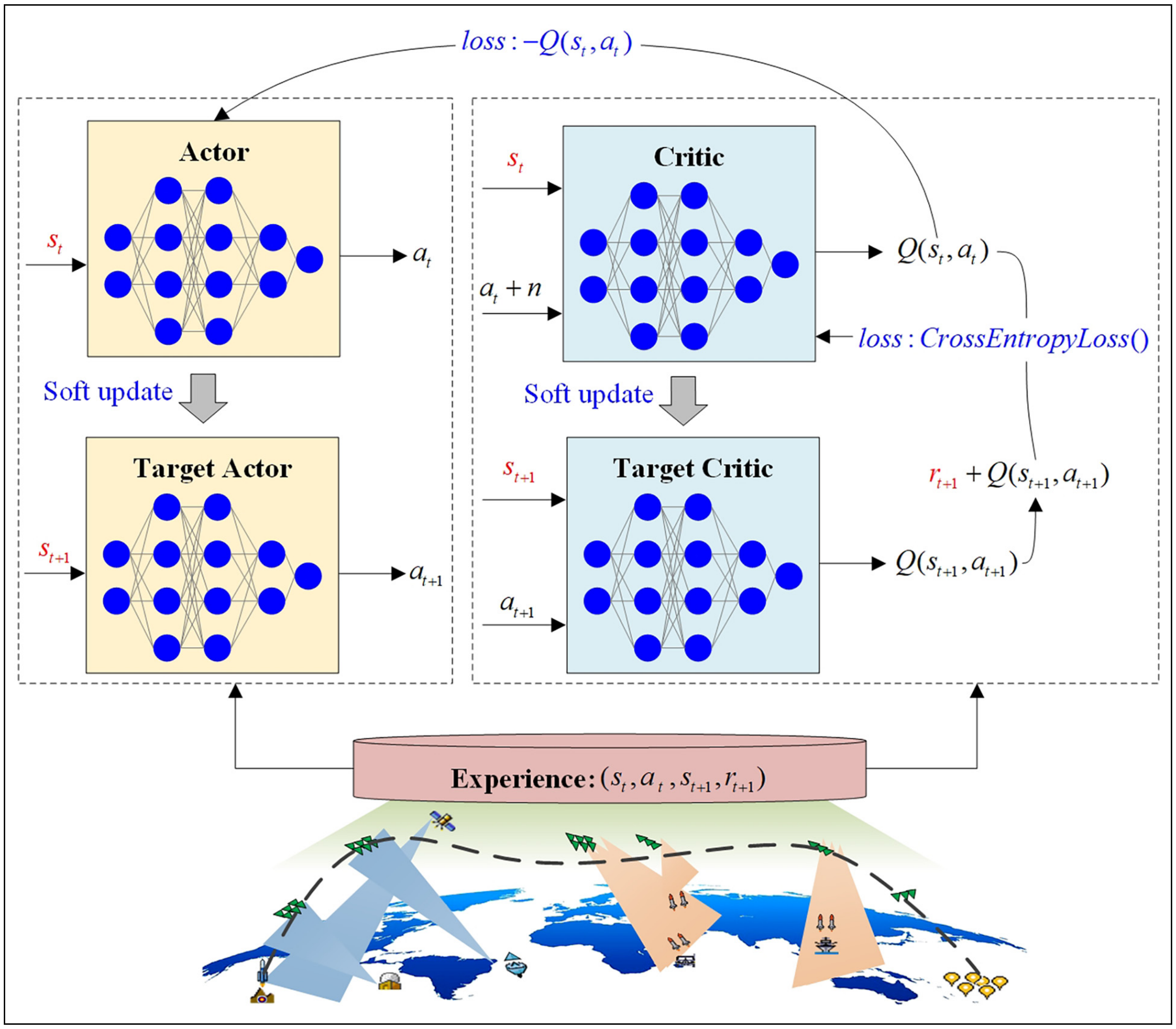

Figure 10. The model structure of DDPG-based multi-target assignment decision.

added. The objective function for obtaining the optimal strategy is formulated as follows

$$\pi^* = \arg\max_{\pi} \underset{\tau\sim\pi}{E}\left[\sum_{t=0}^{\infty} \gamma^t (R(s_t,\ a_t,\ s_{t+1}) + \alpha H(\pi(\bullet|s_t))\right] \tag{12}$$

where α is a regularization coefficient used to control the importance of entropy, which is a trade-off between exploration and exploitation. The greater the α, it is more likely for the algorithm to obtain the optimal strategy, but also more difficult for it to converge. Conversely, the algorithm tends to use the current optimal strategy for decision-making, reducing the proportion of random exploration to quickly converge at the risk of easily falling into local optimal solutions.

5. Simulation and analysis

This section presents the design of a set of experimental schemes to conduct simulations. Moreover, the model training process is presented, followed by an investigation of the exploratory data analysis.

5.1. Experimental design

To conduct simulation more efficiently, we set four experimental modes that represent four types of experimental schemes, each with its individual experimental requirements based on different simulation objectives, as listed in Table 4.

For example, when the experimental mode is set as 0, the behavioral script selects the rule-based decision process, and this mode does not generate data samples. This

Table 4. Experimental schemes.

Mode	Name	Decision type	Data samples	Description
0	RULE_NO_SAMPLE	Rule	No	Simulation test
1	RULE_SAMPLE	Rule	Yes	Initialization
2	NN_TRANING	Network	Yes	Iterative training
3	NN_APPLICATION	Network	No	Network application

mode is mainly used to validate simulation models by conducting a simulation; thus, it is not required to call the network and generate unnecessary data samples. Consequently, the computing resources are minimized. After one simulation is validated, the objective of the second mode is network initialization. In this mode, the rule decision is used, and data samples are generated by conducting numerous Monte Carlo simulations. The data samples with a high RTH index are then selected to train an initial network via supervised learning technologies. Based on the initial network, the third mode starts iterative training via reinforcement learning. In this mode, data samples are dynamically generated and saved. Finally, the fourth mode calls the trained network to conduct simulations, and the performance of the network decision is analyzed in comparison with the rule decision.

Considering each experiment, the factors used as inputs include the longitude and latitude of the blue warship, in addition to the pre-defined target assignment scheme. On one hand, the longitude ranged from -142 to -137, and the latitude ranged from 45 to 50, each of which was divided into six levels. On the other hand, there are 720 possible target assignment schemes since the experiment was set as six warheads versus six targets. Five rounds of Monte Carlo simulations were run for each experiment. In particular, there were 25,920 experimental schemes and 129,600 simulations in total, as we selected a full factorial experimental method. In addition, we recorded the RTH index as the experimental response.

After creating the input batch experimental files, simulations were run using two Intel (R) Core (TM) i7-7700 Central Processing Units with 3.6 GHz and 16 GB of Random Access Memory. It required 84 h to execute all the simulations, which generated an output file containing the RTH index for the respective input conditions and target assignment schemes. Based on the RTH index, we obtained improved target assignment schemes valued at $\{85, 124, 223, 271, 432, 689\}$, with each representing the order number in full permutation combination. Moreover, each simulation generated a database file containing four data tables, including the final state, final action, current state, and reward, which correspond to one state transition $\{s, a, s', r\}$ of the Markov decision process. The data samples were selected if the total reward exceeded 100, and the database name was stored in a text file. The results revealed that 5817 databases were selected, which were used to train a supervised learning network in the following step.

5.2. Model training

Before training the network, a train–validate–test split was performed, which allocated 80% of the data to the training set. Subsequently, validation using a 10-fold cross-validation technique[62] was performed to address the overfitting problem after considering the trade-off between computational cost and generalization capability, and 20% of the allocated data were used for the test set. To realize a network with higher accuracy, we applied several widely used supervised learning algorithms to train the same selected databases, including an Artificial Neural Network (ANN), Support Vector Classifier (SVC), Gaussian Naive Bayes (NB), Classification and Regression Tree (CART), k-Nearest Neighbors (KNN), Linear Discriminative Analysis (LDA), and Logistic Regression (LR).

Figure 11 presents the boxplots of the above algorithms and their respective means and standard deviations (std.). As can be seen from this figure, the ANN demonstrated the highest mean score of 0.975 and the lowest standard deviation of 0.038. In the process of ANN training, the network state was saved in intervals of 1000 episodes, and each episode represented the scanning of a database, given that each complete simulation generated a database. The adaptive moment estimation (Adam)[63] optimizer, which is an extremely popular training algorithm for ANN, was employed.

It is necessary for the model to have sufficient capacity to successfully fit the training set. However, overfitting should be avoided. In any case, note that the training loss is less than the validation loss, although not by a large order of magnitude. It is expected that the model would perform better on the training set since the model parameters are being shaped by the training set. However, the main goal is to also reduce both training and validation losses. Although, ideally, both losses would be roughly the same value, as long as the losses are reasonably close, there is still room for the model to improve generalization capability. The training results showed that the training

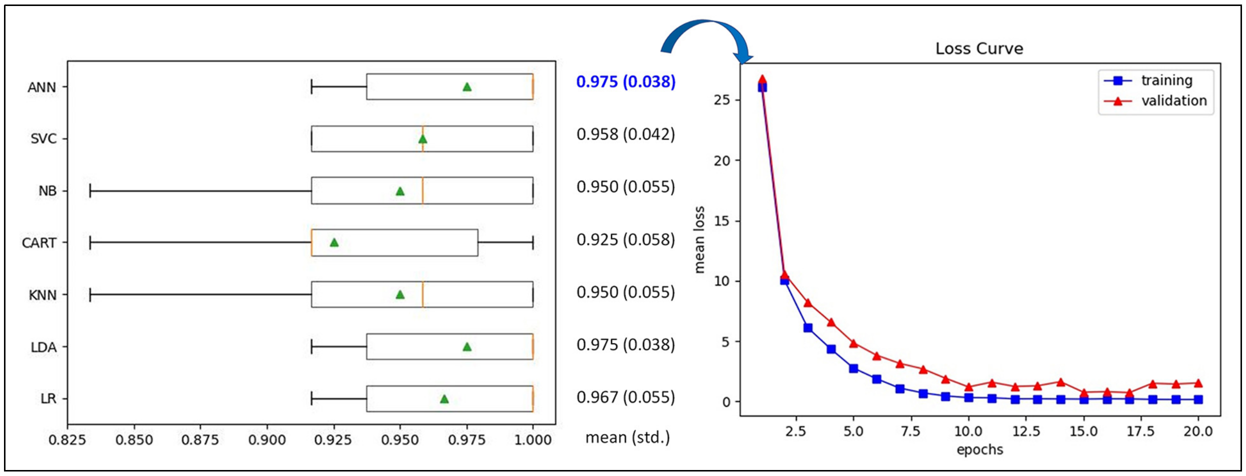

Figure 11. Boxplots of applied supervised learning algorithms.

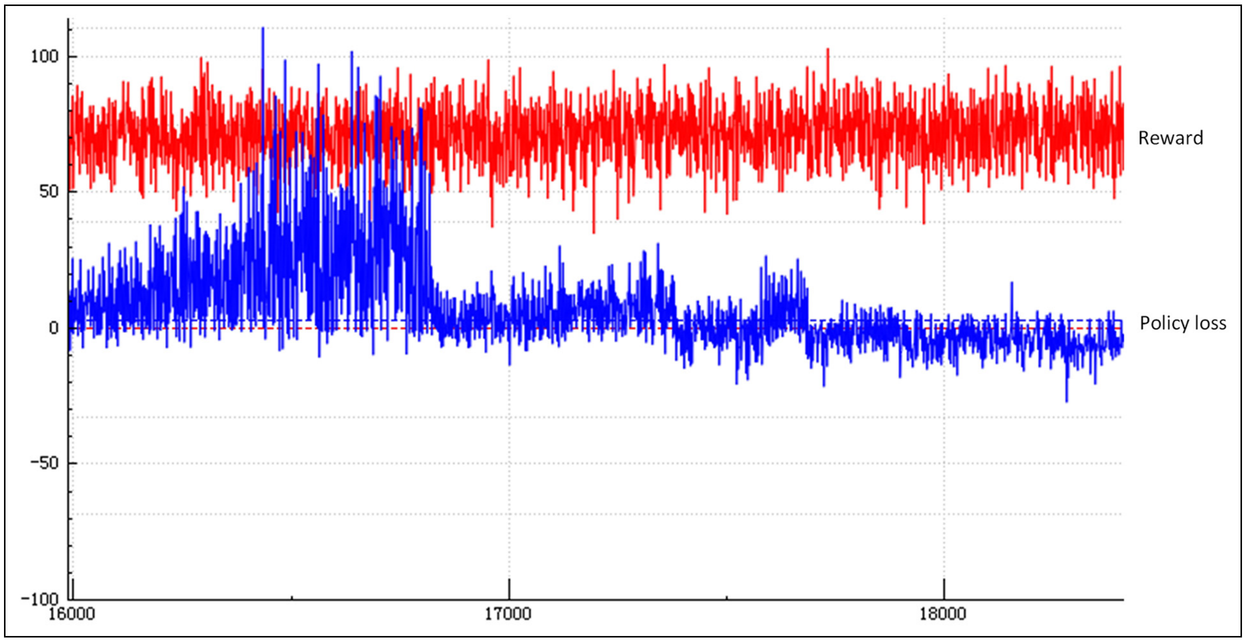

Figure 12. The training state of reinforcement learning.

and validation losses have different absolute values but similar trends, indicating that overfitting is under control and early training and validation termination is acceptable.

This training process required approximately 3.5 h. Moreover, the curve abruptly shifted downward at the start and then stabilized. In addition, the mean training loss value reached the minimum of 0.156 when running 20 epochs. The training process stopped early at 10 epochs with a validation loss of 0.323. It should be noted that we computed a mean loss after each epoch, and the learning rate was set as 0.0001.

When the ANN model was initialized, the iterative training of the subsequent reinforcement learning agent was initiated. This process required approximately 8 h when the policy loss dropped to almost 0, and at this instant, almost 17,000 iterations were conducted, as shown in Figure 12.

In addition, the reward oscillated between 50 and 100. This is because the reinforcement learning agent was performed based on a machine learning agent that was previously trained, so there is little room for additional training. Several critical hyperparameter settings are presented in Table 5.

Table 5. Hyperparameter settings.

Parameter	Value	Description
alpha	0.0003	Learning rate
gamma_actor	0.99	Discount factor_actor
gamma_critic	0.99	Discount factor_critic
tau	0.005	Soft update parameter
size	256	Batch size

Table 6. Descriptive statistics of the Ratio of Target Hits index.

Statistics	Rule	Net
Average	0.414985	0.604537
Median	0.333	0.567
Minimum	0.333	0.45
Maximum	0.5	1

5.3. *Exploratory data analysis*

Regarding the difference between the rule-based and the net-based decisions, exploratory data analysis was initiated with the verification of the main descriptive statistics of their respective RTH indexes, as listed in Table 6. In the case of the rule-based decision, the RTH index ranged from 0.333 to 0.5, with a mean of 0.415 and a median of 0.333. In the case of the net-based decision, the variable ranged from 0.45 to 1, with a mean of 0.605 and a median of 0.567. These results are satisfactory since the RTH index increased with respect to the abovementioned statistics when the net-based decision was used in the same scenario of the engagement simulation. For example, the mean RTH index increased by 18.96% for the net-based decision when compared with the rule-based decision.

A histogram was generated to visualize the distributions of rule-based and net-based decisions, as shown in Figure 13. For the rule-based decision, the data of the RTH index were mainly distributed around the average value of 0.415, whereas the data of the net-based decision were distributed from 0.4 to 1, which is approximately a half normal distribution around the average value of 0.605. Although several simulation experiments in the rule-based case may have better performance than the net-based decision, these occur rarely. For the same number of total simulation runs, the net-based decision has better results statistically than the rule-based decision with respect to the RTH index.

6. Conclusions

There are numerous challenges associated with the application of knowledge-based or data-driven methods to the

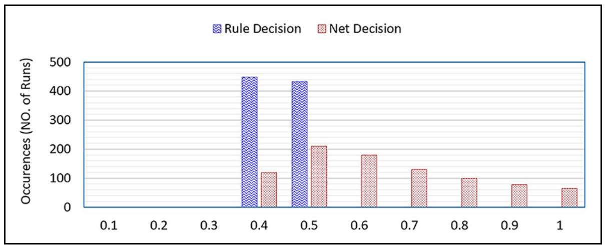

Figure 13. Histogram for rule-based and net-based decision.

representation of complex behaviors in current combat systems. Notably, due to the confidentiality of the military domain, the collection of high-quality data is complicated or otherwise inaccessible, thus limiting the validation and guidance of intelligent algorithms. It is, therefore, a common practice in the military modeling and simulation community to conduct simulations as an effective method for understanding the system behavior or accumulating data samples. Hence, it is critical for simulation modelers to improve the fidelity of knowledge-based models while developing data-driven methods.

In this study, the literature on current modeling techniques used in constructing engagement simulations was investigated in detail. Traditionally, these methods can be roughly classified into two categories, that is, knowledge-based simulation modeling and data-driven data modeling. It should be noted that this does not represent a strict classification. In practice, there is no absolute boundary, given the increasing number of advanced models that use both methods and the rapid development of AI and big data technologies. Beyond the methods described above, the domain knowledge and observed or simulated data can be used synergistically to train intelligent models, that is, the third method of intelligent modeling using the knowledge-based and data-driven methods cooperatively.

For an adequate integration of knowledge and data, this paper presents a discussion on a knowledge-based and data-driven behavioral modeling architecture for engagement simulations. In this architecture, the knowledge of combat systems is loosely decoupled into physical and behavioral parts, such that behavioral models that are developed with Python scripts can be readily re-used, modified, and integrated into physical models. Moreover, the simulations of physical models generate data samples, including data sets for the state, action, and reward, which are then used for the training of smart agents using certain intelligent algorithms. In essence, a smart agent can be viewed as a black box or function with an internal computing logic that does not require physics-based equations. In particular, only the inputs and outputs require consideration. For example, it represents a decision point of the overall process of behavior, and the decision logic is hidden and inexplicable. It should be noted that this proposed architecture is a method for intelligent behavioral modeling benefiting from the two-fold application of knowledge-based and data-driven abstractions. Additional methods are discussed in the related work section.

We then developed a novel modeling formalism referred to as the FDT. The objective was to enhance the degree of modularity of behavioral models, thus decreasing their complexity. Hence, behavioral models built by FDT are represented in the form of a cascaded tree structure, such that the decision process can be decoupled at different levels of decision points. In particular, several decision points that are described by rules can be conveniently replaced with off-line trained smart agents. In addition, we detailed an experimental framework for how to carry out the tasks of model training and test, including several workflows such as the initial experiment, ML, DRL, and comparative experiments. A benefit of using this framework is that the DRL agent is trained on the ML agent; thus, fewer iterations are required for DRL training.

As a proof of concept, we illustrated a target assignment case of ballistic missile penetration. Based on the proposed methodology, we described the combat environment and detailed the ballistic missile model, including the static structure, physical computation, and behavioral representation using FDT. A smart DRL agent was designed for the target assignment decision, including the DNN architecture and the applied DDPG algorithm. Simulation results revealed that the smart agent-based decision demonstrated superior engagement effectiveness to the traditional rule-based decision. This case study can serve as a reference for projects aimed at the development of intelligent capabilities in various engagement simulation systems. However, some effort for tailoring should be required.

Future research should focus on the investigation of improved mechanisms for the use of knowledge and data in the context of engagement simulation. In addition, improvements in the architecture used for the DNN should be investigated with respect to improving the results and efficiency, that is, with a lower computational cost in the training process. To further validate and enhance the adaptability of a smart agent, more combat scenarios with different initial conditions are required to achieve desired warfighting mission effects under the guidance of mission engineering.[64] Also, more advanced simulation models of ballistic missiles may be used in the future to increase the reliability of results.

Acknowledgements

The authors would like to acknowledge the anonymous referees for their reviews and all the volunteers who provided helpful comments on previous versions of this document. Special thanks to the Weapon Effectiveness Simulation System (WESS) team which is directed by Prof. Yonglin Lei for providing the simulation platform.

Funding

This work was supported by the National Natural Science Foundation of China (grant number 62003359) and the National Defense Pre-Research Foundation of China (grant number 80901020104).

ORCID iDs

Zhi Zhu https://orcid.org/0000-0003-3758-8568
Hessam Sarjoughian https://orcid.org/0000-0002-1326-9485

References

1. Cao L. A new age of AI: features and futures. *IEEE Intell Syst* 2022; 37: 25–37.
2. Gill SS, Xu MX, Ottaviani C, et al. AI for next generation computing: emerging trends and future directions. *Internet Things J* 2022; 19: 1–34.
3. Abdelmegid MA, O'Sullivan M, González VA, et al. A case study on the use of a conceptual modeling framework for construction simulation. *Simulation* 2022; 98: 433–460.
4. Seo KM, Choi C, Kim TG, et al. DEVS-based combat modeling for engagement-level simulation. *Simulation* 2014; 90: 759–781.
5. McGraw RM and MacDonald RA. Abstract modeling for engineering and engagement level simulations. In: *Proceedings of the 2000 winter simulation conference*, Orlando, FL, 10–13 December 2000, pp. 326–334. New York: IEEE.
6. Ernest N, Carroll D, Schumacher C, et al. Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions. *J Def Manag* 2016; 06: 1–7.
7. Pope AP, Ide JS, Mićović D, et al. Hierarchical reinforcement learning for air combat at DARPA's AlphaDogfight Trials. *IEEE Trans Artif Intell* 2022; 1: 1–15.
8. Kim BS, Kang BG, Choi SH, et al. Data modeling versus simulation modeling in the big data era: case study of a greenhouse control system. *Simulation* 2017; 93: 579–594.
9. Bocciarelli P, D'Ambrogio A, Falcone A, et al. A model-driven approach to enable the simulation of complex systems on distributed architectures. *Simulation* 2019; 95: 1185–1211.
10. Finegan DP, Zhu J, Feng XN, et al. The application of data-driven methods and physics-based learning for improving battery safety. *Joule* 2021; 5: 316–329.
11. Ng MF, Zhao J, Yan QY, et al. Predicting the state of charge and health of batteries using data-driven machine learning. *Nat Mach Intell* 2020; 2: 161–170.
12. IEEE Computer Society. 1516-2010 —IEEE standard for modeling and simulation (M&S) high level architecture (HLA)—framework and rules, http://ieeexplore.ieee.org/servlet/opac?punumber=5553438 (2010, accessed 21 November 2022).
13. He Q, Zhang M and Gong J. An introduction of BOM modeling framework. *IJMLC* 2011; 1: 353–358.
14. ESA. SMP 2.0 handbook (issue 1 revision 2) EGOS-SIM-GEN-TN-0099, https://taste.tuxfamily.org/wiki/images/9/9a/SMP_2.0_Handbook_-_1.2.pdf (2011, accessed 21 November 2022).

15. Peterson JL. *Petri net theory and the modeling of systems*. 1st ed. Upper Saddle River, NJ: Prentice Hall, 1981.
16. Gill A. Introduction to the theory of finite-state machines. *Math Comput* 1962; 92: 63–74.
17. Schaad R. Parallel functional decision trees for situated agent control. *Dr. Dobb's J* 1999; 24: 62–70.
18. Azar MC. *Assessing the treatment of airborne tactical high energy lasers in combat simulations*. Master's Dissertation, *Air Force Institute of Technology*, Dayton, OH, 2003.
19. Miller JO, Jason L and Honabarger B. Modeling and measuring network centric warfare (NCW) with the system effectiveness analysis simulation (SEAS). In: *Proceedings of the 11th ICCRTS coalition command and control in the networked era*, Dayton, OH, March 2006, pp. 1–20.
20. Chen YZ and Zhang P. Modeling and simulation oriented to the multi-military mission of U.S. army. *J Comm Control* 2018; 4: 89–94.
21. Huang J, Zhao X, Hao JG, et al. Brief introduction of KD-HLA: an integrated environment to support M&S based on HLA. In: *Proceedings of the second international conference on computer modeling & simulation*, Sanya, China, 22–24 January 2010, pp. 281–283. New York: IEEE.
22. Zhu Z, Lei YL and Zhu YF. Model-driven combat effectiveness simulation systems engineering. *Defence SCI J* 2020; 70: 54–59.
23. Brandstein A and Horne G. Data farming: a meta-technique for research in the 21st century (Maneuver Warfare Science). Technical report, Naval War College, Newport, RI, 1998.
24. Sanchez SM. Data farming: methods for the present, opportunities for the future. *ACM Trans Model Comput Simul* 2020; 30: 1–30.
25. Feldkamp N, Bergmann S and Strassburger S. Knowledge discovery in simulation data. *ACM Trans Model Comput Simul* 2020; 30: 1–25.
26. Barry P, Zhang JP and McDonald MM. Architecting a knowledge discovery engine for military commander utilizing massive runs of simulations. In: *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining*, Washington, DC, August 2003, pp. 699–704.
27. Fisher RA. The use of multiple measurements in taxonomic problems. *Ann Eugen* 1936; 7: 179–188.
28. Quinlan JR. Introduction of decision trees. *Mach Learn* 1986; 1: 81–106.
29. Pineda FJ. Generalization of back-propagation to recurrent neural networks. *Phys Rev Lett* 1987; 59: 2229–2232.
30. Burges CJC. A tutorial on support vector machines for pattern recognition. *ACM T Intel Syst Tec* 1998; 2: 121–167.
31. Li D. A tutorial survey of architectures, algorithms and applications for deep learning. *APSIPA Signal Inf Process* 2014; 3: 1–29.
32. Paternina-Arboleda CD and Das TK. A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. *Simul Model Pract Theory* 2005; 13: 389–406.
33. Kessler C, Capocchi L, Santucci JF, et al. Hierarchical Markov decision process based on DEVS formalism. In: *Proceedings of 2017 winter simulation conference*, Las Vegas, NV, 3–6 December 2017, pp. 1001–1012. New York: IEEE.
34. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015; 518: 529–533.
35. Lillicrap TP, Hunt JJ, Pritzel A, et al. Continuous control with deep reinforcement learning. In: *Proceedings of 4th international conference on learning representations*, San Juan, Puerto Rico, May 2016, pp. 1–14.
36. Hasselt VN, Guez A and Silver D. Deep reinforcement learning with double Q-learning. In: *Proceedings of the thirtieth AAAI conference on artificial intelligence*, Phoenix, AZ, 2016, pp. 2094–2100.
37. Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In: *Proceedings of the 33rd international conference on machine learning*, New York, 2016, pp. 1928–1937.
38. Schulman J, Levine S, Moritz P, et al. Trust region policy optimization. In: *Proceedings of the 32nd international conference on machine learning*, Lille, France, 2015, pp. 1889–1897.
39. Harrnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: *Proceedings of the 35th international conference on machine learning*, Stockholm, 2018, pp. 1861–1870.
40. WSC. Exploring big data through simulation, https://pubsonline.informs.org/do/10.1287/LYTX.2014.05.12/full (2014, accessed 21 November 2022).
41. Tolk A. The next generation of modeling & simulation: integrating big data and deep learning. In: *Proceedings of the conference on summer computer simulation*, San Diego, CA, July 2015, pp. 1–8.
42. Afram A and Shari F. Review of modeling methods for HVAC systems. *Appl Therm Eng* 2014; 67: 507–519.
43. Thierry AS, Bastien P, Vittori E, et al. ''Smart entity''-how to build DEVS models from large amount of data and small amount of knowledge. In: *Proceedings of simulation tools and techniques*, Chengdu, China, August 2019, pp. 615–626.
44. Bae KH, Mustafee N, Lazarova-Molnar S, et al. Hybrid modeling of collaborative freight transportation planning using agent-based simulation, auction-based mechanisms, and optimization. *Simulation* 2022; 98: 753–771.
45. Cristea A and Okamoto T. Knowledge computing method for enhancing the effectiveness of a WWW distance education system. In: Brusilovsky P, Stock O and Strapparava C (eds) *Adaptive hypermedia and adaptive web-based systems*. Berlin: Springer, 2000, pp. 2–5.
46. Routray A, Osuri KK, Pattanayak S, et al. Introduction to data assimilation techniques and ensemble Kalman filter. In: Mohanty UC and Gopalakrishnan SG (eds) *Advanced numerical modeling and data assimilation techniques for tropical cyclone prediction*. Berlin: Springer, 2016, pp. 307–330.
47. Xie X. Data assimilation in discrete event simulations. PhD Thesis, Delft University of Technology, Delft, 2018.

48. Zhu Z and Lei YL. Model & data hybrid driven smart modeling for combat systems. In: *Proceedings of the 2020 summer simulation conference*, San Diego, CA, November 2020, pp. 1–12.
49. Laroque C, Skoogh A and Gopalakrishnan M. Functional interaction of simulation and data analytics-potentials and existing use-cases. In: *Proceedings of simulation in produktion und logistik*, Kassel, 2017, pp. 403–412.
50. Saadawi H, Wainer G and Pliego G. DEVS execution acceleration with machine learning. In: *Proceedings of the symposium on theory of modeling & simulation*, Pasadena, CA, April 2016, pp. 1–6.
51. Deist T, Patti A, Wang Z, et al. Simulation assisted machine learning. *Bioinformatics* 2018; 35: 1–11.
52. Zhang HX, He BS, Lu GY, et al. A simulation and machine learning based optimization method for integrated pedestrian facilities planning and staff assignment problem in the multimode rail transit transfer station. *Simul Model Pract Theory* 2022; 115: 102–449.
53. Shao G, Shin SJ and Jain S. Data analytics using simulation for smart manufacturing. In: *Proceedings of the 2014 winter simulation conference*, Savannah, GA, December 2014, pp. 2192–2203.
54. Grieves MW. Product lifecycle management: the new paradigm for enterprises. *Int J Prod Dev* 2005; 2: 71–84.
55. Glaessgen E and Stargel D. The digital twin paradigm for future NASA and U.S. Air Force vehicles. In: *Proceedings of the 53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference*, Honolulu, HI, April 2012, pp. 1818–1832.
56. Zhu Z, Lei YL, Sarjoughian H, et al. UML-based combat effectiveness simulation system modeling within MDE. *J Syst Eng Electron* 2018; 29: 1180–1196.
57. Schadd M, Sternheim AM, Blankendaal R, et al. How a machine can understand the command intent. *JDMS*. Epub ahead of print 5 August 2022.
58. Strembeck M and Zdun U. An approach for the systematic development of domain-specific languages. *Software Pract Exper* 2010; 39: 1253–1292.
59. Goodfellow I, Bengio Y and Courville A. *Deep learning*. Cambridge, MA: MIT press, 2016.
60. Cohen J, Cohen P, West SG, et al. *Applied multiple regression/correlation analysis for the behavioral sciences*. London: Routledge, 2013.
61. Liu CS. Ornstein–Uhlenbeck process, Cauchy process, and Ornstein–Uhlenbeck–Cauchy process on a circle. *Appl Math Lett* 2013; 26: 957–962.
62. Hyndman RJ and Koehler AB. Another look at measures of forecast accuracy. *Int J Forecast* 2006; 22: 679–688.
63. Kingma DP and Ba JL. Adam: a method for stochastic optimization. In: *Proceedings of the 3rd international conference on learning representations*, San Diego, CA, May 2015, pp. 1–15.
64. Garrett RK, Fairbanks JP, Loper ML, et al. The application of applied category theory to quantify mission success. *Simulation* 2023; 99: 201–220.

Author biographies

Zhi Zhu is an Assistant Professor at the National University of Defense Technology, China. He was a visiting PhD student at Arizona State University from 2016 to 2017. His research interests are model-driven engineering, systems simulation, and machine learning.

Tao Wang is an Associate Professor at the National University of Defense Technology. He is the director of the Department of Strategy Management Engineering. His research interests include strategy planning, multi-agent simulation, and data mining.

Hessam Sarjoughian is an Associate Professor at the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University, USA. He is the co-director of the Arizona Center for Integrative Modeling and Simulation. His research interests are model composability, simulation-based design, and agent-based simulation.

Weiping Wang is a Professor at the National University of Defense Technology. He is a senior fellow of China Simulation Federation. His research interests are simulation optimization, intelligent decision, and knowledge computing.

Yuehua Zhao is an Associate Professor at the Nanjing University, China. She received her PhD degree from the University of Wisconsin–Milwaukee, USA. Her research interests are data mining, knowledge management, scientometrics.