IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 November 2022, accepted 28 November 2022, date of publication 8 December 2022,
date of current version 14 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3227797

APPLIED RESEARCH

Experimental and Computational Study on the Ground Forces CGF Automation of Wargame Models Using Reinforcement Learning

MINWOO CHOI[1], HOSEOK MOON[1], SANGWOO HAN[2], YONGCHAN CHOI[3], MINHO LEE[1], AND NAMSUK CHO[1]

[1]Department of Operation Research, Korea National Defense University, Nonsan 33021, South Korea
[2]Agency for Defense Development, Daejeon 34186, South Korea
[3]Department of Economics and Management, Korea Army Academy, Yeongcheon 38092, South Korea

Corresponding author: Namsuk Cho (ncho64@gmail.com)

This work was supported by the Agency for Defense Development by the Korean Government under Contract UE202075ID.

ABSTRACT Wargame is an important tool that enables training units to develop various strategies by allowing them to experience unexpected situations. There are three methodologies that determine the behavior of the Computer Generated Forces(CGF) in wargame—rule-based, agent-based, and learning-based methodologies. The military determines the behaviors of the CGF mainly based on the rules because a doctrine and an operation plan are well established. However, the advent of intelligent weapons and the accompanying changes in tactics will make it difficult to expect an environment and situations of the future battlefield. Therefore, we studied the automation of CGF through reinforcement learning in order to give unexpected situations, so that the training unit would be able to establish various strategies and tactics through the wargame model. Based on the combat functions of the ground forces, we configured multiple environments that the ground forces CGFs will learn in. First, infantry and artillery CGFs learned in the close combat environment, which is the basis of ground forces combat. Second, the trainee CGF learned in the context of military training. Third, the drone CGF learned how to reconnaissance and attack in a multi-drone environment, and finally, the combat service support CGF learned under the mission of supplying ammunition. As a result, we confirmed that the reinforcement learning methodology is applicable to CGF through these experiments.

INDEX TERMS Wargame, CGF, ground forces combat, reinforcement learning, artificial intelligence.

I. INTRODUCTION

Wargame has been proved to be a valuable tool for understanding the uncertainty of the battlefield and the changing paradigm of war, especially in terms of training troops and verifying operational plans [1]. As a result, the role of wargame has become more critical in the future battlefield environment, which is changing rapidly.

A. IMPORTANCE OF AUTOMATING CGF

Computer Generated Forces (CGF) describes the behavior of combatants or weapons systems in the wargame model [2], and CGF's behavior is essential in simulating a realistic battlefield environment and combat situation [3], [4], [5]. One of the most critical roles of CGF is to make training participants perceive CGF to be realistic, and therefore, it is necessary to automate CGF because of the following reasons. First, in the current wargame model, CGFs do not work well in an undefined situation, because they behave under predefined conditions such as military doctrine, operation plan, and simulation logic of existing models. Second, outcomes of the wargame may be affected by a difference in the ability of the gamers controlling CGF, even though only the operation plan established by the Operation Planning Process(OPP), the Course Of Action(COA) of the commander, and staff should be the factors determining the success or failure of the wargame. Third, the atypical training situation in wargame caused by the automation of CGF can enhance the wargame's

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

immersion of wargame and increase the combatants' training effect [6]. Last, automating CGF (a) reduces the operational requirements of game operators, (b) increases the efficiency of troop operations, and (c) reduces administrative requirements, such as inputting a scenario [7].

B. CGF AUTOMATION METHODOLOGY AND REINFORCEMENT LEARNING

There are three representative methods of automating CGF: rule-based, agent-based, and learning-based.

Rule-based is a methodology in which CGF operates according to defined rules. This methodology makes the behavior of CGFs intuitive and easy to implement, since the military writes manuals or plans. However, it is difficult to express all situations as a rule, and gamers who understand the rule of CGF operation can perform actions which are not possible in the real battlefield to win the wargame. For example, when troops are trained through actual wargame, the gamers of the troop can easily understand the rules of the opposing forces CGF and use tactics to win the game, because the rules of behavior of CGF implemented in wargame are relatively simple. For this reason, the effectiveness of training is sometimes reduced [7].

The agent-based methodology provides minimal rules to agents and determines behaviors through mutual information exchange. Agent-based methods have fewer rules to be defined than rule-based methodologies, in that they can express undefined behavior through information exchange between agents. In Context-Based Reasoning (CxBR) and Belief-Desire-Intention (BDI), which are two representative methods of the agent-based wargame model, all goals and actions of the agent must be defined as scenarios. When creating a scenario, domain knowledge must be communicated well for the agent to function properly. However, as the size of the unit grows, scenario creation becomes more complex, which makes it difficult to implement the scenario into a model [8].

Learning-based methodologies use machine learning, such as supervised, unsupervised, or reinforcement learning. This method best expresses the cognitive ability of combatants, in that it can autonomously judge agents' behavior. Although there is a study of training two combatants CGF with a supervised learning [9], supervised and unsupervised learnings are not widely used, as they require a large amount of data. In addition, it is difficult to judge whether a certain combat activity is right or wrong by using those learning methods [10]. On the other hand, reinforcement learning has the advantage in that it does not require correct answers, as agents build data by repeating episodes in the environment and improve behaviors in correct directions through rewards. As agents can learn various tasks even in a multi-agent environment, the description of CGF using reinforcement learning is becoming a reality [11], [12].

So far, we have described three methodologies for CGF automation. Since the military possesses well established doctrines and operation plans, it mainly uses the rule-based methodology in the wargame model and has shown its effectiveness in combat readiness. Nevertheless, it is necessary to convey CGF research with reinforcement learning, as expressing every situation as a rule is impossible, and it is cumbersome to define a rule and input it into the wargame every time a new weapon system or tactic comes out. The agent-based method can be thought as an alternative, but this method requires defining rules as much as the rule-based method does, along with writing a battle scenario. Therefore, we studied the automation of CGF by using a reinforcement learning method that does not require separate rules and scenarios. By applying this advanced artificial intelligence technologies, we expect to find improvements in CGF.

Our study aims to create a simulation environment in consideration of combat situations, and make CGF learn by reinforcement learning algorithms to confirm the automation possibility of ground forces CGF. While setting environments, we considered the general situations of the military, such as close combat, a reconnaissance operations, and logistic support missions. The structure of this study is as follows: in Section II, we described the literature review of CGF automation by applying the three methodologies described above. Section III describes the environment established to use the reinforcement learning methodology, and Section IV presents experimental results of this study. Finally, in Section V, we present conclusions.

II. LITERATURE REVIEW

This section describes CGF automation research by using rule-based, agent-based, and reinforcement learning methodologies.

In rule-based methodology, CGF determines the correct behavior by taking into account the real-time situation of the model through predefined rules. However, it is difficult for CGF to decide on the suitable action as military knowledge and manuals are vast, and a variety of situations can occur on the actual battlefield. Since the military has well-established rules such as doctrine and operational plans, it is more important to study the framework that allows the CGF to determine the action by efficiently processing the rules rather than to study the rules themselves. As a framework study, there are studies the operation of CGF on the framework by modularizing the battle decision process and the algorithm that can operate in the framework [13], [14]. In addition, the rule-based system has a limitation in that it dose not consider the uncertainty of the battlefield. In [15], the CGF acted by using a sequential decision-making model that probabilistically considers uncertain or partially observable situations.

In the agent-based methodology, agents decide their actions through information exchange with other agents under minimal rules. Therefore, agent-based studies include organizing agents as well as communication of agent-to-agent information well. For example, in the study of an agent management, a small combat model framework under a multi-agent environment was established by configuring agents that determine major tasks, and sub-agents that

perform functions such as maneuver and detection [16]. And in [17], the researcher created a virtual command center agent to induce cooperation of agents, so as to make CGFs cooperate under a multi-agent environment. Some countries in NATO are studying agent based models by applying CxBR and BDI methods. The idea of CxBR is that humans use only a fraction of their knowledge when they infer, by classifying situations faced by agents into context and making a limited choice over one of the actions that fit the context. BDI is another methodology that describes human reasoning methods. Belief represents information about the agent's situation, and Desire means the goal that the agent should achieve. Intention stands for the process of the agent selecting an action by considering the Belief and the Desire. Huet al. [8] compared of the performances of the C2(Command and Control) system in cases when CxBR and BDI were respectively applied to it. There are other CGF studies done with CxBR and BDI, such as [18] and [19].

CGF automation using reinforcement learning methodology focuses mainly on Air Force fighters as CGF. The researchers automated the fighter's behavior in a two-to-one aerial combat situation through reinforcement learning [20]. In the works in [21], researchers compared the performance of trained fighter CGFs and human-manipulated CGFs. In this study, trained CGF lost to skilled pilots but won against beginner-level pilots, suggesting the possibility of the CGF automation using reinforcement learning methodology. In addition to the combat behavior of the fighter CGF, there was a study done to learn maneuver. The researchers in [22] studied the maneuver of fighter CGF in an environment where an Anti-Aircraft Defense system exists, by using a curriculum learning of making CGF reach the goal while solving sub-problems. Also, reinforcement learning research on ground forces CGF compared algorithms in a simple environment considering Rendezvous with obstacle avoidance [23].

CGF research using reinforcement learning also operates under a casual game environment. For example, the agent learned from the classic game Pong or Breakout from the Atari2600 game environment [24]. Some studies have been conducted in 3D environments that are more complex than 2D environments. For example, in Doom, a 3D-based FPS (First Person-Shooting) game, the agent learned how to shoot. As a result, the trained agent performed better than the rule-based agent and the human-manipulated agent [25]. Whereas the previous two studies have only considered a single agent, and had relatively simple environments, some studies such as Google DeepMind's case of StarCraft reinforcement learning were done under more challenging conditions [26], [27]. Despite a complex environment of StarCraft in which all settings change in real-time and multiple agents have to choose actions, CGF has succeeded to learn. In addition, individual units of StarCraft have learned with multi-agent algorithms. StarCraft research has been continued, and the current level of learning is to a degree of winning against professional gamers.

Applying reinforcement learning to CGF for wargame differentiates from using reinforcement learning on casual game players, as wargames and casual games have different purposes and methods. Casual games aim to make players complete the final mission, but wargames aim to make them establish various tactics and strategies by providing an indirect experience of the combat situation of the training unit. For this reason, there are differences in the way the casual game and war game progress. Casual games are linear and game progress step-by-step, in that a player completes the final mission while solving sub-missions with correct answers. On the other hand, there is no right answer in wargames, as the behavior of the CGF can vary depending on the tactics used even in the same battle situation.

A. THE CONTRIBUTION OF RESEARCH

The main contribution of our research, which applies reinforcement learning for the automation of CGF in the wargame model, is that we experimented with various combat environments after creating environments based on the combat functions of the ground forces. The details are as follows.

- First, we applied reinforcement learning to automatically simulate the CGF of the ground forces wargame model. Until now, most of the cases of simulating the wargame's CGF as reinforcement learning were made for the Air Force's fighters. Although there are previous studies on CGF of ground forces, but most of them were conducted by (a) explaining the necessity of applying reinforcement learning methodologies, (b) theoretically predicting the effects without specific experiments when applying the reinforcement learning method (c) or relatively simplifying the composition of objects and environments.
- Second, our study reflects the various characteristics of the ground forces. It is difficult to apply the fighter CGF research to the ground forces because the combat of the ground forces is based on the Combined Arms Combat (CAC) in which two or more branches cooperate. In addition, the process of describing ground combat is more complex than that of describing air combat. Moreover, combat functions such as maneuver, intelligence, and logistics are systematically performed in ground combat, making it necessary to model and test these combat functions. In order to fulfill this condition, we have tested the implementation of CAC, multi-drones, and CSS(Combat Service Support) environments.
- Finally, we can suggest the applicability of reinforcement learning CGF to wargame models from an expert's point of view. To apply reinforcement learning CGF to the wargame model, the experimental results must be analyzed from the perspective of the military. Our study verified whether the ground forces CGF behaves correctly based on a high level of understanding in the military field by comparing it with its doctrine.

III. METHODOLOGY

This section describes the environment for experiments and defines states, actions, and rewards according to the environment.

A. EXPERIMENTAL DESIGN

We considered the combat function of the ground forces to create a wargame environment. Combat function refers to the military roles and activities that must be performed to achieve the concept of operational execution in ground operations. Also, it consists of six functions: command and control, intelligence, maneuver, protection, firepower, and logistic support. Ground forces must perform and integrate these functions when conducting operations to exert combat power. Therefore, we configured the experimental environment as follows.

There are four experimental environments for reinforcement learning of ground forces CGF: close combat, military training, reconnaissance drones, and logistic support. Table 1 shows the details of each environment. First of all, close combat is the situation that is the most frequently encountered by ground forces on the battlefield. A battle between infantries and a battle under artillery fire support are the examples. Therefore, we composed the close combat environment for the following four cases, while assuming situations that could take place on the battlefield.

- Case1: Case1 is a 1:1 infantry combat environment where a single infantry agent is trained. The purpose of this agent's learning is to learn how to recognize the opponent as an enemy and attack them to win the combat. The experimental environment is shown in Figure 1(a).
- Case2: Case2 is a 2:2 infantry combat environment where infantry agents learn to cooperate. Cooperation between combatants is essential in real combat because a squad consists of several combatants.
- Case3: Case3 is a 2:2 environment where infantry teams fight against artillery teams. Both Infantry and artillery CGF, as agents, are expected to learn to combat differently reflecting the characteristics of different weapons they have. To this end, we made the firing range and firing interval of the artillery CGF to be three and five times longer than that of the infantry CGF, respectively.
- Case4: Case4 is a 2:2 environment in which one infantry and one artillery fight as a team, and the team's CGFs are expected to learn cooperative combat between agents of different branches in this environment. This cooperation is called Combined Arms Combat, and it increases the power of the team. Figure 1(b) visualized this experimental environment.

While Case1 to Case4 dealt with a maneuver and a firepower among the combat functions of the ground forces, the following three cases consist of environments which are associated with intelligence, logistics, and military training.

- Case5: Case5 is an environment set to make trainee CGF learn. Although the combat behavior specified in the manual does not necessarily guarantee making optimal choice, it is still necessary to train trainee CGF as specified in the doctrine. Trainee CGF has to engage the enemy and reach the goal. The purpose of this learning is that trainee CGF would do both things—and therefore, it will not consider reaching the target without engaging the enemy. The Figure1(c), (d) shows the environment.
- Case6: Case6 is an environment for learning drone CGFs. Automation research on drone CGF is essential, because drones are highly valuable as reconnaissance and attacking weapons. Basically, the leader drone and the follower drone learn how to find and attack the enemy within the operational area, while the follower drones additionally learn to maintain a certain distance from the leader drone during the fight. Figure 1(e) shows the environment of Case6, with the drone group consisting of 1 leader drone and 5 follower drones, and the target group consisting of infantry and tanks.
- Case7: In Case7, we trained two CSS CGFs. Combat Service Support (CSS) is a battlefield function which is vital for combat continuity. The CSS CGF's task is to deliver ammunition according to the principle of logistics: to deliver the right amount of ammunition, at the right time, and to the support units that need it the most. Figure 1(f) shows the environment of Case7, consisting of two CSS CGFs carrying ammunition, Ammunition Supply Point (ASP), and five units requiring ammunition supply. The environment of Case7 is different from the general delivery service environment, in that the supply amount and supply location were not planned in advance.

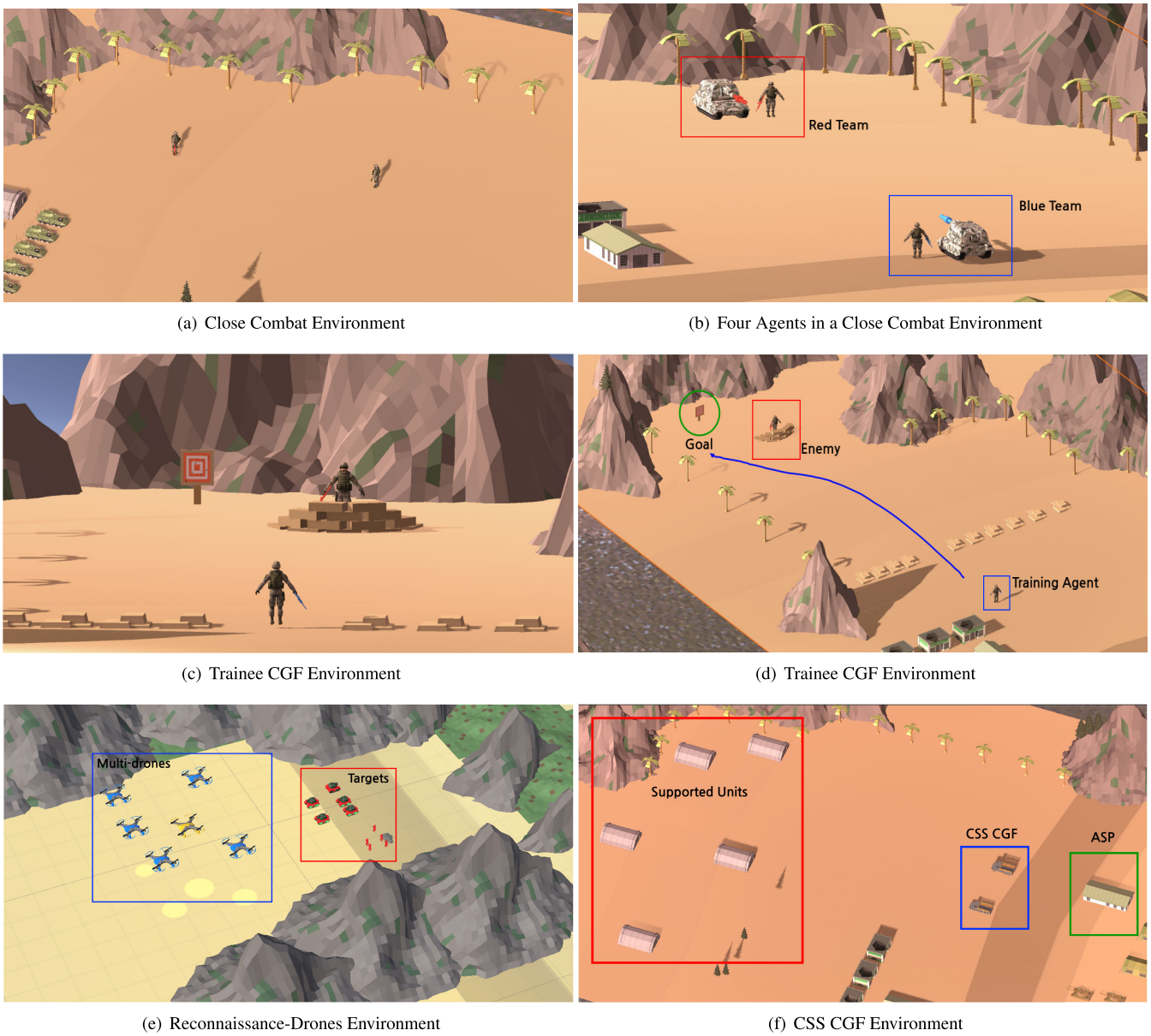
B. TOOLS AND ALGORITHMS

Configuring the environment is important in reinforcement learning, as it determines the learning direction of CGF. There are two main ways of creating the environment in reinforcement learning. One is to use open-sources that provide pre-built environments. For example, OpenAI Gym, Gym Roboschool, Gym Extensions, or PyBullet can be considered [28], [29]. The other way is for researchers to use general programming languages such as Python, Matlab, or Net-Logo [28], [30]. We chose the Unity [31] to construct 7cases of Section III-A. It's best to use currently used wargame as an environment, but unfortunately, there is no wargame model that supports reinforcement learning. In such an aspect, research using Unity has the following advantages. First, Unity can describe a variety of environments, including physical laws, so Unity can implement CGF and resolutions most similar to wargame model. Second, Unity is a highly reliable simulation tool, as it has been used widely in various fields. Last, Unity provides the latest reinforcement learning package that researchers can use with convenience, thereby reducing administrative requirements for organizing experiments.

The following is information about the algorithms used in our study. Research on reinforcement learning can be

TABLE 1. Experimental design.

Environment(task)	Closed Combat				Military Training	Recon Drones	Logistics Support
Case	Case1	Case2	Case3	Case4	Case5	Case6	Case7
Agent Type	Single Agent	Multi Agents			Single Agent	Multi Agents	Multi Agents
Description	Infantry VS Infantry	2Infantries VS 2Infantries	2Infantries VS 2Artilleries	Infantry Artillery VS Infantry Artillery	Trainee CGF	1 Leader Drone 5 Follower Drones Targets	2 CSS CGF 1 ASP 5 Supported Units
Trainee Agents	Infantry	2Infantries	2Infantries 2Artilleries	Infantry Artillery	Trainee CGF	1 Leader Drone 5 Follower Drones	2 CSS CGF
Algorithms	PPO & Self-Play	MA-POCA & Self-Play			PPO	MA-POCA	MA-POCA
Purpose of learning	Enemy recognition	Cooperation	Reflect weapon characteristics		Multi-mission	Detect and attack enemies within communication range	Ammunition supply



(a) Close Combat Environment

(b) Four Agents in a Close Combat Environment

(c) Trainee CGF Environment

(d) Trainee CGF Environment

(e) Reconnaissance-Drones Environment

(f) CSS CGF Environment

FIGURE 1. Experimental environments.

usually divided in two parts; first, the part of improving the performance of algorithms, and second, the part of applying algorithms to solve problems under a specific domain. The purpose of this study is consistent with the second reason.

Therefore, we use the following representative algorithms that are proven to be fair. The algorithm we used for each case is shown in Table 1.

- Proximal Policy Optimization (PPO) is based on an Actor-Critic algorithm [32], and it shows an excellent performance as an algorithm that increases learning speed through importance sampling techniques and data reuse. More detailed information can be found on [33].
- Multi-Agent Posthumous Credit Assignment (MAPOCA) is a multi-agent algorithm. As multi-agents are generally rewarded at a group level, all agents will be rewarded if an agent team wins the game. To reward in such a way, centralized critiques, which are a neural network that serve as a 'coach' for the entire group of agents, were trained and improved to even reward agents who did not directly contribute to the victory. Further details of the algorithm are provided on [34].
- Self-play is a method that increases the effectiveness of learning in environments of competing with opponents, such as chess, tennis, and soccer. It is selectively applicable to PPO and MA-POCA algorithms, which are used in our study [35], [36].

In our experimental environment, we used the PPO algorithm when learning a single agent and the MA-POCA algorithm when learning multiple agents. In addition, in an environment where agents of the same type battle each other, a self-play algorithm was additionally applied, and the algorithm used in each case is shown in Table 1.

C. STATE, ACTION, REWARD

State is information observed by the agent, and the agent determines the action by considering the state. For example, in an environment where infantry CGF learns, a researcher can set the location of enemies and terrain as state information, and movement and attack as actions. Next, the reward is the value the agent receives from the environment when the agent selects an action. Because agents update policy in the direction of maximizing reward, researchers control the behavior of CGF through reward. For example, the researcher gives infantry CGFs a positive reward if they win a battle with an enemy and a negative reward if they lose. For the experiment, the state, action, and reward for each environment are set as follows.

1) State

In our study, CGF collects state information using Unity's radar function called Raycast. S is the state space of the environment. O is the observation space of all radars $O{:=}O^1 \times \ldots \times O^n$ where O^i denotes the observation space of radar $i(i{=}1,2,\ldots,n)$. At time t, $o_t^i \in O^i$ is the local observation of radar i which is correlated with $s_t \in S$. Finally, B_j is the binary variable which indicates whether object $j(j{=}1,2,\ldots,m)$ is detected or not.

$$B_j = \begin{cases} 1, & \text{If the object j is detected by radar} \\ 0, & \text{otherwise} \end{cases}$$

Therefore, the local observation can be defined as $o_t^i{=}(B_1, B_2, \ldots, B_m, H, D)$ where H is binary variable indicating whether any object is hit by radar i. D is distance between radar i and object hit by radar i. For instance, if the nth radar's local observation at time t is $o_t^n{=}(1,0,0,1,5)$, then the first object at distance 5 has been detected by nth radar. The collect information meaning CGF state information for Cases1 to 7 is shown in Table 2.

TABLE 2. State description for each case.

Case	CGF(agent)	Collect information
Case1 to 4	Infantry, Artillery	Enemy Friendly forces Wall
Case5	Trainee	Goal, Enemy
Case6	Leader Drone, Follow Drone	Enemy Friendly forces
Case7	CSS	ASP Supported Unit

2) Action

At time t, we define action vector $a_t = (a_t^1, a_t^2, \ldots, a_t^l)$ where a_t^k is the kth action the agent can take $(k{=}1,2,\ldots, l)$. Also, actions that can be taken are divided into three types: continuous, discrete, and binary. If the action is continuous, a_t has a real value between 0 to 1; if it is binary, a_t has a value of 0 or 1. The defined actions of the agents in our environment are shown in Table 3. For example, in Case1 of Table 3, if $a_t = (0.7, 0.9, 0.1, 0)$ at time t, it signifies that the agent moves vertically by 0.7, horizontally by 0.9, and rotates by 0.1×360 degrees, while the agent does not carry out the attack.

TABLE 3. Action description for each case.

Case	CGF(agent)	Type	Actions
Case1 to 4	Infantry Artillery	continuous	Vertical Horizontal Rotate Attack
Case5	Trainee	continuous	Vertical Horizontal Rotate Attack
Case6	Leader Drone Follower Drone	discrete	Left, Right Up, Down Attack
Case7	CSS	discrete	Left, Right Up, Down Rotate

3) Reward

We set all values for rewards empirically. Also, two considerations were taken into account when we decided under what circumstances to give the rewards and how much the rewards should be. First, the

definition of a reward must conform to the doctrine; and second, the rewards should conform to the reward policy of other researchers.

Cases1 to 4 are the environments where the episode ends with a positive reward for winning and a negative reward for losing. Case1 is a single agent environment where the agent receives $+1$ rewards if it wins and -1, if loses. Cases2 to 4 are multi-agent environments, and therefore, we configured an additional reward for the attack behavior, in order to trigger the combat behavior of the agents on the team. In Cases2 to 4, the reward function for the reward, which is denoted by r can be viewed in Equation (1).

$$r = \begin{cases} 1, & \text{count(enemy forces)} = 0 \\ -1, & \text{count(friendly forces)} = 0 \\ 0.1, & \text{agent attack the enemy} \end{cases} \tag{1}$$

where count(agent) means the number of agents in the environment.

In Case5, rewards are given according to the importance of the task that the agent must perform. It gives a high reward for reaching the goal, which is to accomplish an important mission, while giving a relatively low reward for the mission to attack the enemy. Also, the reward is determined by calculating the distance between two objects in the reward function, where the distance $d(a, b)$ is defined as the Euclidean distance between a and b. The reward function of Case5 is defined as Equation (2).

$$r = \begin{cases} 1, & d(\text{trainee, goal}) = 0 \\ -1, & d(\text{trainee, goal}) = 0 \wedge \text{count(enemy)} = 1 \\ -1, & \text{moving outside the environment} \\ 0.1, & \text{agent attack the enemy} \end{cases} \tag{2}$$

In Case6, the team rewards received by the leader drone and the follower drone are the same. The team reward is given when the drones find or attack a target, and the rewards differ depending on the type of a target. Additionally, an additional individual reward will be given to the follower drone according to its distance from the leader drone. Through this, each follower drone will be able to maintain the communication distance with the leader drone. Equation (3) shows the reward function of Case6.

$$r = \begin{cases} 10, & \text{count(targets)} = 0 \\ 2, & \text{eliminates(or search) a tank} \\ 1, & \text{eliminates(or search) an infantry} \\ 0.001, & d(\text{leader, follow}) < m \end{cases} \tag{3}$$

where m is desired operating distance between leader and follower drone.

In Case7, we define U_{capacity} as a variable indicating the remaining amount of ammunition of a unit. CSS CGFs obtain a reward when supplying ammunition to a unit, and the size of the reward is determined based on this variable, as shown in Equation (4). According to the reward structure, CGFs gain a large reward when U_{capacity} is small, so that they would serve ammo with consideration over the remaining ammunition of each unit. In other words, CGFs are expected to prioritize the allocation to units that lack the ammunition.

$$r = \begin{cases} 20, & U_{\text{capacity}} < 20 \\ 10, & 20 \leq U_{\text{capacity}} < 40 \\ 1.5, & 40 \leq U_{\text{capacity}} < 60 \\ 0.5, & 60 \leq U_{\text{capacity}} < 80 \\ 0.1, & 80 \leq U_{\text{capacity}} \\ -1, & \text{out of area} \end{cases} \tag{4}$$

where a unit's ammo amount ranges from 0 to 100.

D. HYPER-PARAMETERS

This section describes the hyperparameters shown in Table 4. Since selecting hyperparameters affects the algorithm performance considerably, it is common to find the optimal hyperparameters through a sensitivity analysis [37]. However, since our study aims on applying the algorithm, rather than improving it, we used the hyperparameters value of other studies that are similar with ours.

TABLE 4. Hyperparameters description for each case.

Hyperparameters	Case1,5,7	Case2,3,4	Case6
batch_size	128	2048	1024
buffer_size	2048	20480	10240
beta	0.005	0.005	0.01
epsilon	0.2	0.2	0.2
lambda	0.95	0.95	0.95
gamma	0.99	0.99	0.99

Hyperparameters consist of the following five elements. First, batch_size indicates the number of experiences (size of data) in each iteration of gradient descent search, while buffer_size denotes the number of experiences that should be collected before updating the policy model. Also, beta is a value for the randomness of the policy and increasing it will result in an increased number of random actions. Epsilon is a value that indicates how rapidly policy evolves. For instance, if epsilon is small, it means that the policy is stable, but the training process would be slow. Finally, lambda is the value showing how much of the previous reward was reflected when evaluating the future reward, and gamma is the discount rate for the future reward.

IV. RESULTS

This section describes the experimental results of the 7 cases presented in Section III. Although we have presented only a few important graphs in analyzing the results in the main text,

all graphs can be viewed in Appendix A. Our experiment was performed in a computer environment with Apple M1 chip's 8-core CPU, 7-core GPU, and 8GB of RAM.

A. CASE1 AND CASE2

In Case1 and Case2, we trained infantry CGF—Case1 was a 1:1 environment, and Case2 was a 2:2 environment. The detailed results are as follows.

- Case1: CGF was successfully trained in the Case1 environment, where CGF battles against an agent with CGF's previous policy, as shown in Figure 1(a). An interesting observation is that the reward value converges to a certain number, which is 0.3 in this case. It is able to understand the meaning of the number through knowing that Case1 is a zero-sum environment. Case1 is the combat situation with a clear winner and loser. Therefore, if the agent wins all battles, the reward will converge to 1; however, if the agent keeps losing, reward will approach to -1. 0.3 is a number derived when the agent wins 6.5 out of 10 times and loses 3.5 times; in other words, when the win rate is about 65%. Also, since the values show the clear convergence, it was concluded that 65% is the best policy found by the agent in the current experimental environment.

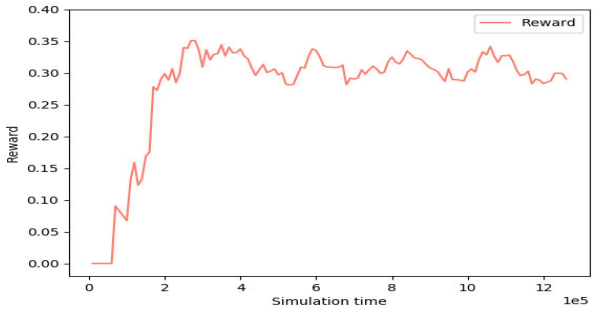


FIGURE 2. Dynamic of reward for Case1.

- Case2: This experiment, which aimed to make CGFs learn how to cooperate, was successful; we observed the CGFs trying to occupy an advantageous position over the enemy in combat. To evaluate the effectiveness of the model learning cooperation, 3000 battle simulations were performed against the CGF trained in Case1. As a result, the agent in Case2 won 2160 times, which means that the win rate was 72%. From this number, we were able to estimate the benefits of the cooperation.

B. CASE3 AND CASE4

In Cases3 and 4, artillery is newly joined as an agent to test the learning of agents with different classes or weapon systems. In Case3, we organized infantry CGF and artillery CGF into different teams, while in Case4 we set infantry CGF and artillery CGF in the same team. The results of the experiment are as follows.

- Case3: In Case3, a team of two infantry and a team of two artillery fought against each other. As the learning progressed, we could observe the pattern of artillery with a long weapon range fighting from a distance, while infantry tried to fight at a close distance. This is because infantry CGFs learned that it is the most profitable for them to approach the artillery CGF for a combat, while the artillery CGFs learned that the optimal policy for them is to fight while maintaining distances from the infantry CGFs. Figure 3 shows the dynamics of rewards during the learning. As Self-Play was applied in a zero-sum environment, the reward increased when the CGFs trained, while it decreased when they did not. For instance, the infantry team could not receive a reward at the beginning of the learning, because they did not train. However, after the learning began, the reward value started to increase. Figure 4 shows how the agents learn by applying Self-Play.

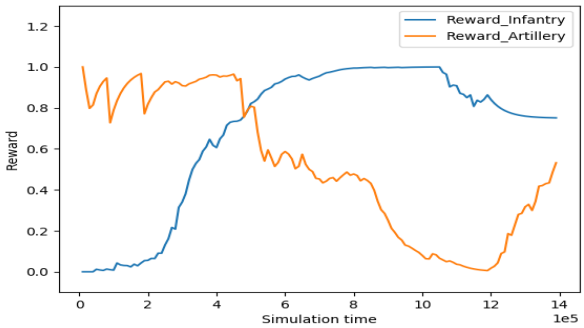


FIGURE 3. Dynamic of reward for Case3 (Blueline indicates Artillery team, Redline indicates Infantry team).

In Step1 of Figure 4, the infantry team learns the policy, and the artillery team performs random actions as part of the environment. In Step2, the infantry team acts as part of the environment with policy #1 learned in Step1, and the artillery team begins learning the policy. CGF repeats this process until the simulation ends.

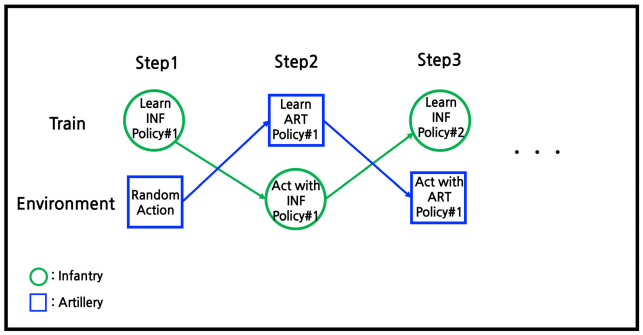


FIGURE 4. Self-play operation procedure.

- Case4: In this experiment, which aimed to confirm the combat behavior of CGFs when infantry and artillery CGFs are a team, CGFs learn cooperative battle by adopting the characteristics of weapons. Infantry CGF carried out close combat under the fire support of

artillery CGF, which corresponded with the actual tactics of infantry on the battlefield. Also, the artillery CGF learned the behavior of fire support, which corresponds with the actual artillery's roles. Although this experiment only reflected the property of weapons in CGF, it has its meaning in showing that CGFs are able to implement doctrinal features.

C. CASE5

In Case5, the agent had a main mission to arrive at the goal. In the process of conducting the mission, an accidental situation, that is the appearance of an enemy, occurred. Therefore, the agent learned how to deal with contingencies, so that it could complete the main mission. Therefore, what we expected to see from the result was that the agent has learned these behaviors by exploring under circumstances without any interference. However, as a result, the agent ended the episode without reaching the main goal, despite it successfully eliminated the enemy. We attributed that the problem was that the agent did not acquire rewards often, which is called "the sparse reward problem."

The sparse reward problem indicates the condition of an agent in reinforcement learning receiving only a sparse reward signal from the environment, so that it will be difficult for the agent to connect the signals with future rewards. This problem can reduce the learning speed or rate. In order to solve the sparse reward problem, the researcher should take steps that allow the agent to receive the reward better.

Although the representative method for this is Curriculum Learning, it was difficult for us to solve our problem using this method, because it makes CGF learn to arrive at the goal while solving sub-problems. However, if the CGF only learns to reach the target after the process of eliminating the enemy (left in Figure 6), it loses the opportunity to learn how to get to the goal directly without eliminating enemies, that may be the better move than the former depending on the situation (As shown on the right in Figure 6).

From the point of view of the military, CGF must be able to decide whether to eliminate the enemy or not, which is the process that depends heavily on the distance between the goal and the enemy. Therefore, we reorganized the environment, so that CGF can first learn the important things and then learn additional situations.

In the new environment, Case5 is divided into 3 steps, and the environment of the next stage is called when when learning in each steps is completed. Sequentially, the procedure is as follows: in Step1, CGF learns to get to the goal; in Step2, it randomizes the position of the goal; and in Step3, it creates an enemy. Through this step-by-step action of adding or changing the environment, the agent was able to learn the new policy while maintaining the existing policy. Figure 5 shows the experimental results after reconfiguration of the environment. The red line in Figure 5 shows the reward after the reconfiguration, and it shows the significant improvement compared to the previous learning which is shown through the blue line. Point (B) and (C) on the graph shows when

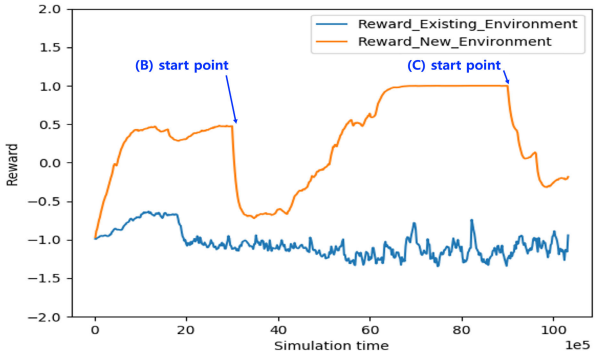


FIGURE 5. Dynamic of reward for Case5 after customized (Blue indicates the existing environment, red indicates the reconstructed environment).

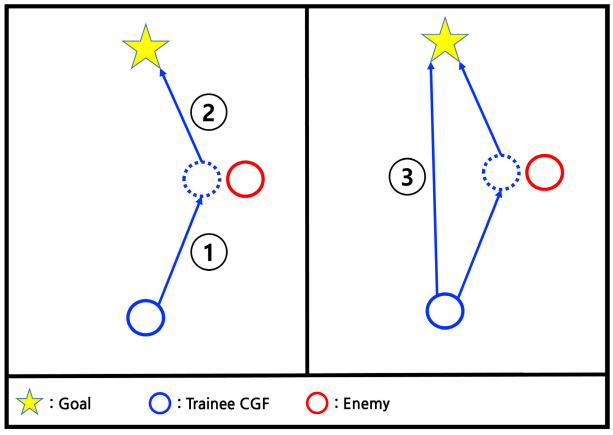


FIGURE 6. Differences between Curriculum Learning(Left) and Our Learning(Right).

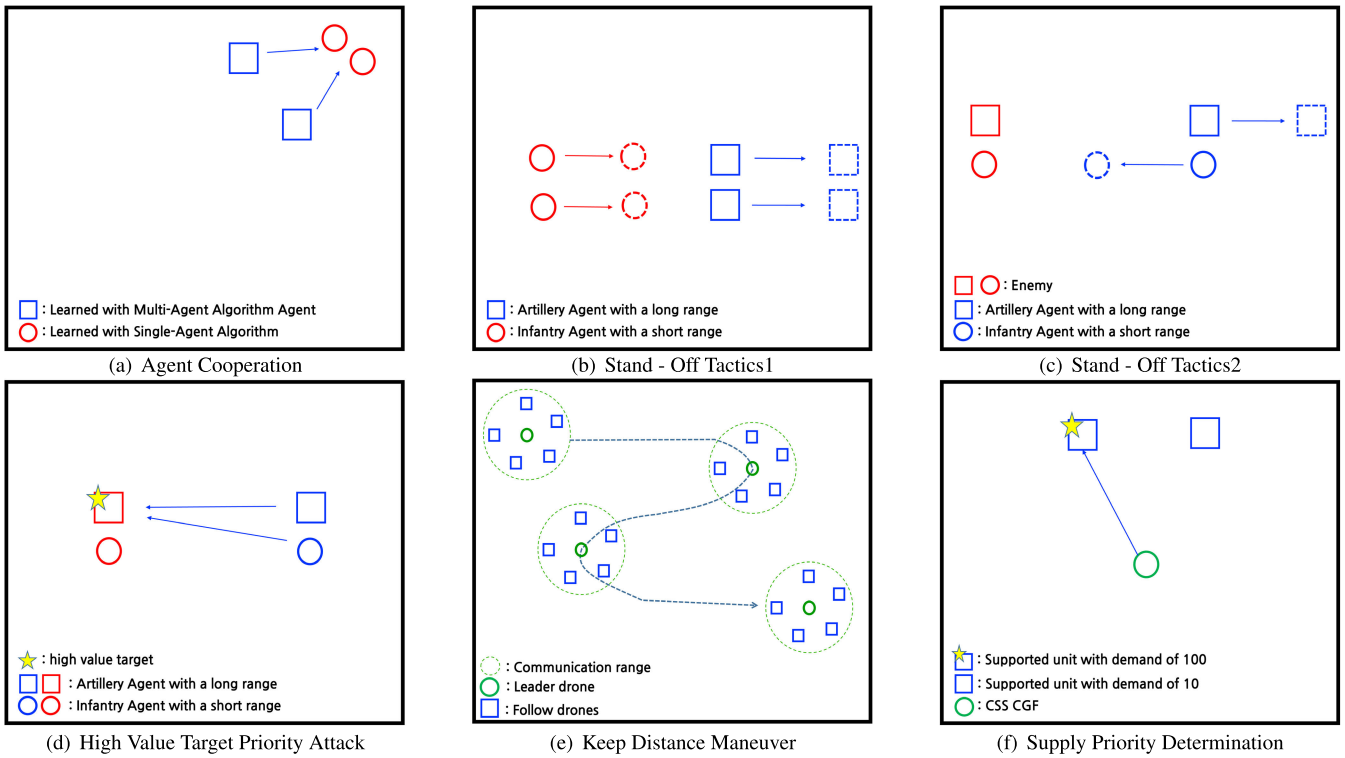
Step2 and Step3 started respectively; therefore, it is revealed the reward decreased significantly at the start points and then it increases again.

D. CASE6

In this experiment, which aimed to eliminate all enemies through the collaboration between drones, the drones have learned the optimal policy. From Case6, we could describe three interesting observations. First, the follower drones learned to find the enemy while they maintained the communication distance with the leader drone. Second, the drones first identified the tank with the higher reward value. Third, drones made the decision of what suits the most for the team's interests between detecting and attacking actions. For instance, we could observe that when the number of all drones was reduced to two, the remaining two drones performed reconnaissance rather than high-risk attacks, to increase survivability.

E. CASE7

In Case7, CSS CGF aimed to supply ammunition to the units by considering the ammunition amount of the units that change in real time, in order that the amount for each will not get exhausted. At the beginning of the learning, CGF supplied



(a) Agent Cooperation (b) Stand - Off Tactics1 (c) Stand - Off Tactics2

(d) High Value Target Priority Attack (e) Keep Distance Maneuver (f) Supply Priority Determination

FIGURE 7. Summary of key experimental results.

ammunition only to nearby units. The reason was because CGF received the same amount of the reward by supplying ammunition to the units regardless of the distance. Hence, CGF focused on the behavior of supplying itself, rather than considering the supply amount of each unit. To solve this problem, we derived a new reward structure as shown in Equation (4) described in Section III-C, that made the amount of ammunition supplied to be more important.

F. OBSERVATIONS

Through the experiment results, we have discovered several remarkable points that show us the possibility of the intelligence of CGF. Although some parts of the observations include the above, they were reinterpreted in consideration of military use in wargames.

1) First, with a multi agent algorithm, CGF learned to cooperate for the team's victory. Figure 7(a) shows CGFs that learned with a multi-agent algorithm (blue square) driving other CGFs that learned with a single agent algorithm (red circle) into a corner, so that the former could prevent the latter from escaping.
2) Second, CGF can learn combat skills without rules. In the close combat experiment, CGFs learned to engage in range combat by reflecting the weapon properties. In Figures 7(b) and (c), CGFs performed a close combat and a Stand-Off battle by considering the range of weapons.
3) Third, CGFs learned that the most appropriate action for the team's victory is to first remove the opponent's high-value target. Figure 7(d) shows team blue's winning strategy in an environment where infantry and artillery are on the same team. The infantry and the artillery increased the chance for winning by attacking the red team's artillery first.
4) Fourth, from a military perspective, it is necessary for CGF to learn under artificially intervened experimental environments. This is because it is not easy for CGF to learn in an environment that requires completing multiple tasks. Therefore, although there are other ways to solve this issue, it is most appropriate for CGF to be trained under the environment that changes step by step as suggested in Case5, when it comes to the military training of CGF.
5) Fifth, in the multi-drone environment of Case6, the follower drones searched for the targets while keeping operation distance from the leader drone, as shown in Figure 7(e). Although the agents had not provided with the rule that the leader drone and the follower drones should maintain a certain distance, they automatically started to operate in this way, as we set maintaining distance as the condition of the reward. This is because the drones learned that this method allows them to maximize the search range while they could maintain the proper distance from each other.
6) Last, in the military support environment of Case7, CSS CGFs determined the priority of supply and started supplying ammunition first to the prioritized place. Figure 7(f) is a diagram that shows how the CGF

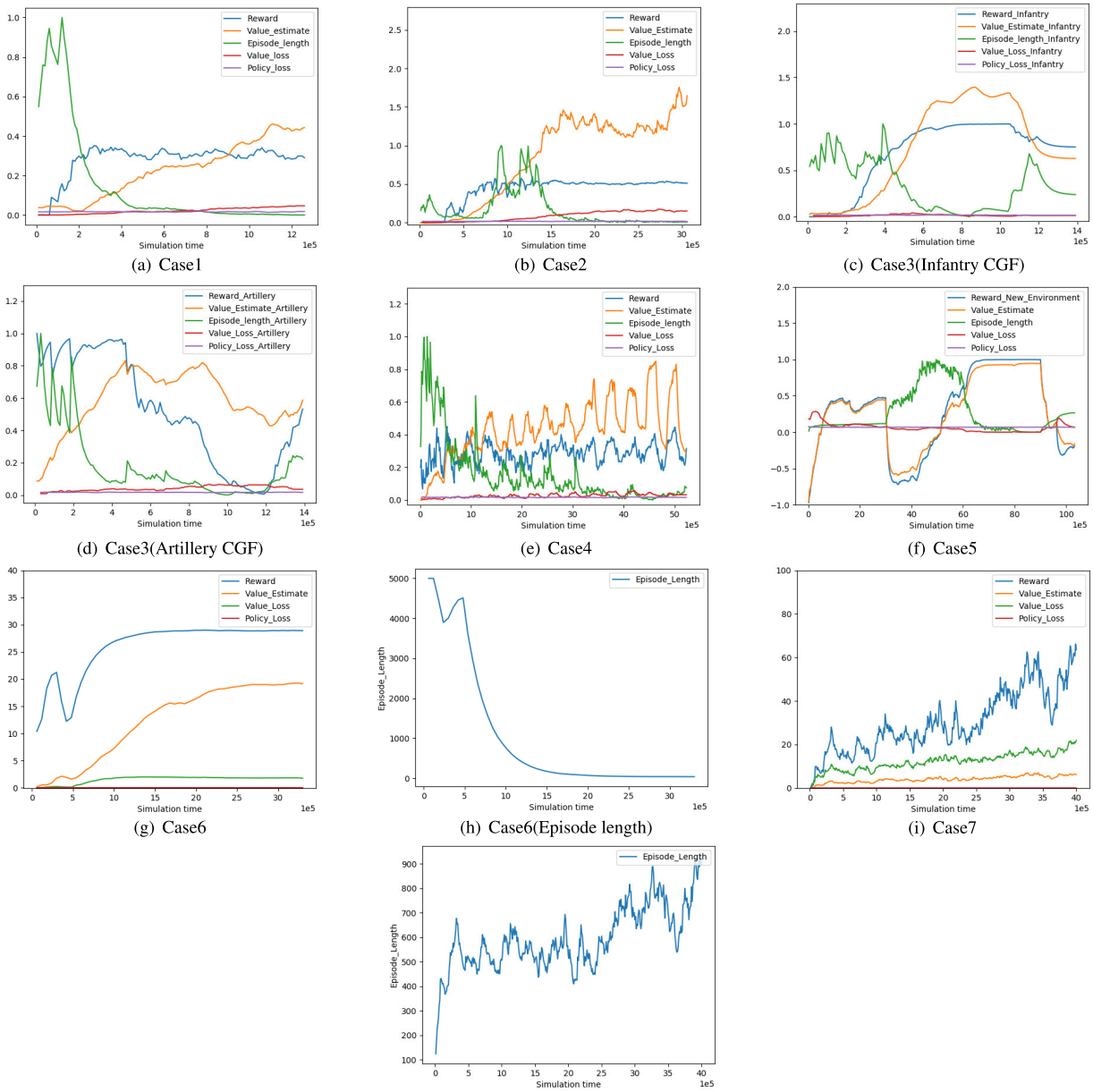


FIGURE 8. Dynamic of results for all Cases.

decided which unit should be prioritized by checking the ammo amount of the units. Although we had not provided CGF with a rule of how it should set supply priority, but it made a correct judgement by itself as a reaction to the reward.

G. FINDINGS

In this section, we described our experimental findings on the automation potential of CGF and proposed several important findings.

1) First, the following experimental results suggest the possibility of applying reinforcement learning CGF to wargame models. (a) CGFs acted to 'maneuver' in the enemy's direction and project 'firepower' to the enemy to win the combat. b) Drone CGFs implemented the 'intelligence' function and reconnoitered high value targets first. (c) CSS CGFs implemented the 'combat support' function. (d) Infantry CGFs and artillery CGFs performed cooperative battles with actions that are suitable to their weapon characteristics, which demonstrates the possibility of cooperation between various branches.
2) Second, the CGF implements doctrinal behavior without rules. In the close combat environment, CGF performed actions such as close combat, Stand-Off tactics, and high-value target identification,

although we have defined only 4 types of actions and 3 types of rewards. This suggests the possibility in the future that various doctrinal behaviors of CGF can be implemented in more complex environments, with only action and reward definitions and without rules.

3) Third, implementing the cooperative behavior of CGF by setting a group reward is possible. CGFs in the experiment performed combat actions that benefit the group, through the multi-agent algorithm. As CGFs prioritized actions that benefit the team over ones that benefited the individuals' interests, there is a possibility for ground forces CGFs to engage in cooperative combat.
4) Fourth, we can expect that reinforcement learning will enable CGF to behave beyond human thinking. The combat actions of the ground forces are established as doctrines and operational plans which were made based on war history and battle simulations; in other words, they are from the human's intelligence. Also, in our study, we considered behaviors of CGF to be correct if they complied with the doctrine. However, in some cases, a behavior that differentiates from the doctrine might not be wrong. CGF is capable of performing actions that have not occurred in the human thought process through accumulating numerous experiences from the learning process. Therefore, we can even expect that in the future behaviors of CGFs which are learned through reinforcement learning will improve the doctrine.
5) Finally, this study has a value as basic research. Although our experiment was not done with the actual wargame model, and thus the research results must be confirmed before they are applied to the existing wargame model, we suggest the possibility of applying reinforcement learning for the automation of the ground force CGF, as we used a verified program, and experts in military theory participated in the experiment.

V. CONCLUSION

In this study, we implemented the doctrinal behavior of CGF by applying reinforcement learning in various environments and confirmed the automation potential of CGF. In addition, we were able to learn CGF in an environment with multiple missions by presenting the learning method that can be applied when training agents militarily. CGFs implemented with reinforcement learning can act intelligently because they can make different decisions based on the behavior of the training unit. Accordingly, if we apply the learned CGF to the war game model, the training unit can experience a lot of unexpected situations, and it is expected that various tactics and strategies can be developed through this.

.

APPENDIX A ADDITIONAL GRAPH FOR EACH CASES

In this section, we provided additional figures derived from the experimental results for each case. In analyzing the experimental results, not only reward but also other elements of the experiment such as loss or episode length are important indicators. In Cases1 to 6, the times at which the episode ended gradually decreased as the reward values increased during the learnings, and the loss values were measured to be less than 1, indicating that the learning proceeded normally. However, in Case7, the episode length, and the loss value increased as well as the reward value. The reason for the increase in episode length is that in Case7's environment, the episode ends only when the CGF leaves the area or reaches the set simulation time. In other words, the increase in episode length means that the CGF has served its mission for a long time.

REFERENCES

[1] K. McCabe, *Service Parochialism and the Defense Planning Process: A Case Study of the Title 10 Wargames*. Montreal, QC, Canada: McGill Univ., 2016, pp. 33–53.

[2] L. J. Luotsinen and E. Sjoberg, "The modeling and analysis of computer generated forces representing groups and organizations in military conflict zones," in *Proc. NATO Symp. Transforming Defense Through Modeling Simul. Opportunities Challenges*, 2012, pp. 1–13.

[3] M. Tambe, "Implementing agent teams in dynamic multiagent environments," *Appl. Artif. Intell.*, vol. 12, nos. 2–3, pp. 189–210, Mar. 1998.

[4] U. Dompke, "Computer generated forces-background, definition and basic technologies," NC3A, ORFS Division, The Hague, The Netherlands, Tech. Rep., NC3A, 2003.

[5] R. Jacobs and P. Brooks, "Computer generated forces-future needs," Inst. Defense Analyses Alexandria, VA, USA, Tech. Rep. ADA485093, 2003.

[6] C. Heinze, B. Smith, and M. Cross, "Thinking quickly: Agents for modeling air warfare," in *Proc. Austral. Joint Conf. Artif. Intell.* Berlin, Germany: Springer, 1998, pp. 47–58.

[7] N. Cho, H. Moon, and J. J. Pyun, "The study on CGF behavior modeling methodologies for defense M&S: Focusing on survey and future direction," *J. Korea Soc. Simul.*, vol. 29, no. 2, pp. 35–47, 2020.

[8] R. A. Lovlid, A. Alstad, O. M. Mevassvik, N. De Reus, H. Henderson, B. Van Der Vecht, and T. Luik, "Two approaches to developing a multiagent system for battle command simulation," in *Proc. Winter Simul. Conf. (WSC)*, Dec. 2013, pp. 1491–1502.

[9] S. Sun and J. Pan, "Behavior modeling of computer generated forces," in *Proc. IEEE Symp. Electr. Electron. Eng. (EEESYM)*, Jun. 2012, pp. 78–81.

[10] A. Toubman, J. J. Roessingh, J. Van Oijen, R. A. Lovlid, M. Hou, C. Meyer, L. Luotsinen, R. Rijken, J. Harris, and M. Turcanik, "Modeling behavior of computer generated forces with machine learning techniques, the NATO task group approach," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 1906–1911.

[11] Z. Peng, R. Luo, J. Hu, K. Shi, S. K. Nguang, and B. K. Ghosh, "Optimal tracking control of nonlinear multiagent systems using internal reinforce Q-learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 4043–4055, Aug. 2022.

[12] Z. Peng, J. Hu, Y. Zhao, and B. K. Ghosh, "Understanding the mechanism of human–computer game: A distributed reinforcement learning perspective," *Int. J. Syst. Sci.*, vol. 51, no. 15, pp. 2837–2848, Nov. 2020.

[13] S. C. Tian, F. Sun, Z. Y. Li, and L. S. Hao, "Research on CGF behavior modeling based on rule," *Appl. Mech. Mater.*, vols. 347–350, pp. 3056–3059, Aug. 2013.

[14] S. Wang and Y. Liu, "Modeling and simulation of CGF aerial targets for simulation training," in *Proc. Int. Conf. Comput. Intell. Syst. Netw. Remote Control*, 2020, pp. 1–14.

[15] K.-H. Lee, L. Hee-Jin, and K. Kee-Eung, "A case study on modeling computer generated forces based on factored POMDPs," in *Proc. Korean Inf. Sci. Soc. Conf.* 2012, pp. 333–335.

[16] S. Sin, J. Y. Lee, J. M. Kim, and J. Pyun, "A study of framework for combat effectiveness analysis using multi agent based simulation," in *Proc. Korean Oper. Res. Manag. Sci. Soc. Conf.*, 2017, pp. 3824–3841.

[17] X. Tan, S. Lai, W. Wang, and M. Zhang, "Framework of wargame CGF system based on multi-agent," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2012, pp. 3141–3146.

[18] Z.-T. Hu and J. Liu, "The research of ASW helicopter ACGF construction based on CXBR," in *Proc. Int. Conf. Comput. Intell. Secur. Workshops (CISW)*, Dec. 2007, pp. 132–135.
[19] S. Mukherjee and K. Saroj, "Engineering emergent behavior in computer generated forces (CGF) using multi-agent system,"
[20] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. Van Den Herik, "Rewarding air combat behavior in training simulations," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 1397–1402.
[21] T.-H. Teng, A.-H. Tan, and L.-N. Teow, "Adaptive computer-generated forces for simulator-based training," *Exp. Syst. Appl.*, vol. 40, no. 18, pp. 7341–7353, Dec. 2013.
[22] J. Källstrom and H. Fredrik, "Reinforcement learning for computer generated forces using open-source software," in *Proc. Interservice/Ind. Training, Simul., Educ. Conf.*, 2019, pp. 1–11.
[23] B. Toghiani-Rizi, F. Kamrani, L. J. Luotsinen, and L. Gisslen, "Evaluating deep reinforcement learning for computer generated forces in ground combat simulation," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 3433–3438.
[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[25] G. Lample and S. C. Devendra, "Playing FPS games with deep reinforcement learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.
[26] O. Vinyals, "StarCraft II: A new challenge for reinforcement learning," 2017, *arXiv:1708.04782*.
[27] J. Foerster, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–14.
[28] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*.
[29] C. Erwin and Y. Bai, "PyBullet a Python module for physics simulation for games," *PyBullet*, 2016. [Online]. Available: http://pybullet.org/
[30] M. E. Aydin and R. Fellows, "A reinforcement learning algorithm for building collaboration in multi-agent systems," 2017, *arXiv:1711.10574*.
[31] *Unity*. [Online]. Available: https://unity.com/
[32] R. S. Sutton and G. B. Andrew, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018, p. 257.
[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[34] *ML-Agents V2.0 Release: Now Supports Training Complex Cooperative Behaviors*. Accessed: May 5, 2021. [Online]. Available: https://blog.unity.com/technology/ml-agents-v20-release-now-supports-training-complex-cooperative-behaviors
[35] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017, *arXiv:1712.01815*.
[36] B. Ospanov, "Training intelligent tennis adversaries using self-play with ML agents," M.S. thesis, Dept. Comput. Sci., Nazarbayev Univ., Astana, Kazakhstan, 2021.
[37] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 754–762.

MINWOO CHOI received the B.S. degree in mechanical engineering from the Kumoh National Institute of Technology, South Korea, in 2008, and the M.E. degree in operations research from Korea National Defense University, South Korea, in 2019, where he is currently pursuing the Ph.D. degree in operation and research. In 2008, he is conducting research and work on military M&S, artificial intelligence, and statistical analysis.

HOSEOK MOON received the B.S. degree in chemistry from the Korea Military Academy, South Korea, in 1994, and the M.E. degree in electronic engineering, the Ph.D. degree in industrial engineering, and the Ph.D. degree in statistics from Korea University, South Korea, in 2003, 2006, and 2010, respectively. He is currently a Professor of military science with Korea National Defense University. His research interests include data science, defense M&S, and AI.

SANGWOO HAN received the B.S. degree in computer science and engineering from Chung-Ang University, South Korea, in 2003, and the M.S. and Ph.D. degrees in information and communications from Gwangju Institute of Science and Technology (GIST), South Korea, in 2005 and 2011, respectively. He is currently working as a Senior Researcher with the Agency for Defense Development. His research interests include combat simulation, system effectiveness analysis, and artificial intelligence.

YONGCHAN CHOI received the B.S. degree in chemistry from Korea Military Academy, South Korea, in 2013, and the M.E. degree in military science from Korea National Defense University, South Korea, in 2021. He is currently an Assistant Professor of economics and management with Korea Army Academy, Yeongcheon. His research interests include big data analysis, military operation research, and deep reinforcement learning.

MINHO LEE received the B.S. degree in chemistry from the Korea Military Academy, South Korea, in 2013, and the M.S. degree in operation research from Korea National Defense University, South Korea, in 2022, where he is currently pursuing the Ph.D. degree in operation research. His research interests include data analysis, military operation research, and deep learning.

NAMSUK CHO received the B.S. degree in computer science from the Korea Military Academy, the M.S. degree in operations research from the Air Force Institute of Technology, OH, USA, and the Ph.D. degree in industrial engineering from the University of Wisconsin—Madison, USA, in 2016. He is currently an Associate Professor with the Military Operations Research, Korea National Defense University. His work focuses on optimization and simulation model for various problems in defense and military domain. He has been involved in many research project including Military Medical Planning, Critical Infrastructure Protection, Computer Generated Forces Modeling, and Models for Reconnaissance Drone.