Achieving Intelligent Behavior in Multi-Domain Electromagnetic Warfare Environments Through Neural Network-Informed Search and Self-Play

Michael Ganger, Gerardo Leal
General Dynamics Mission Systems
Fairfax, Virginia
Michael.Ganger@gd-ms.com, Gerardo.Leal@gd-ms.com

ABSTRACT

Effective personnel training and sub-system testing in synthetic warfare environments has achieved incredible levels of realism and fidelity in physics and human-machine interface. However, progress has lagged in constructive behaviors—specifically, in planning and decision-making over long time horizons—with simulations still relying on scripted or rules-based behavior models. In contrast, we hypothesize that "intelligent" constructive behaviors can be realized automatically through maximization of long-term rewards. To test this, we design an implicit approach using Neural Network informed search, trained through self-play without input from a human expert, to estimate the best action for an entity and expected outcome of a scenario. We demonstrate emergent "intelligent" human-level behavior in multi-domain environments with electromagnetic actions and non-trivial collections of entities, with applications to adversarial training simulations.

ABOUT THE AUTHORS

Michael Ganger is a Data Scientist at General Dynamics Mission Systems working on applications of Machine Learning, especially in the areas of Reinforcement Learning and Computer Vision. He has a broad interest in discovering and applying new techniques in mathematics and data science to solve problems.

Gerardo Leal is General Dynamics Mission Systems Engineering Fellow. He leads research on application of deep learning for synthetic data generation on optical, sonar and radar domains. He has spear headed reinforcement learning research for cognitive electric warfare. He has developed faster than real-time detection and tracking systems for physical security video under poor visibility conditions. Conducted research on use of distributed knowledge graph embeddings in disconnected and intermitted tactical network environments. Developed a framework for automatic optimization of deep neural networks for deployment on power constrained edge devices. He has applied unsupervised machine learning for anomaly detection to better predict impending failures in Autonomous Underwater Vehicles as well to detect anomalous maneuvers in satellites. He has developed Deep Generative Adversarial based systems to classify, clean, and compress side scan sonar data. He has also applied deep generative auto-encoders and various deep convolutional architectures to increase classification accuracy under low SNR of radio frequency (RF) signal modulations. Some of his publications include "A Synthetic Augmentations Machine Learning Approach for Superior Target Detection and Recognition in Poor Visibility Conditions", "Early failure and anomaly detection in AUV's through attention based contextual timeseries forecasting", "Advanced Side-scan sonar analytics through Deep Generative Adversarial Networks", "Subsurface Environmental Adaptability Through Multi-Domain Generative Adversarial Networks", "Smart Data Fabric: Distributed, Eventually Consistent and Subscription-Based Data Distribution for Tactical Environments", "Adaptive Space Mapping for Autonomous Mobile Robot Perception" Gerardo holds a BSE in Electrical Engineering and Computer Science, a Baccalaureate in Applied Physics, and a number of Deep Learning Artificial Intelligence, Computer Vision and Reinforcement Learning specializations.

Achieving Intelligent Behavior in Multi-Domain Electromagnetic Warfare Environments Through Neural Network-Informed Search and Self-Play

Michael Ganger, Gerardo Leal
General Dynamics Mission Systems
Fairfax, Virginia
Michael.Ganger@gd-ms.com, Gerardo.Leal@gd-ms.com

INTRODUCTION

The level of fidelity of a simulation-based training environment significantly affects its effectiveness. To date, high fidelity has been achieved in aspects of simulation which do not require models of human behavior. However, realistic models of human decision-making and planning are lacking, and those that do exist often lack flexibility.

A significant challenge in the design of training simulations is realistic modeling of adversarial behaviors. Traditional approaches to solve this problem often rely on human expertise and complex sets of rules, often limiting the scale, realism, and efficacy of a training simulation. In general, this is related to the difficulty in producing *constructive* behaviors. Constructive models of behavior emulate the actions and decisions of a human operating simulated systems (Modeling and Simulation Coordination Office, 2011).

Games provide a unique substrate for demonstrating automated approaches to human-like behavior. Prior work has investigated using Reinforcement Learning (RL) to optimize offensive forces in a combat scenario against a fixed opponent (Boron & Darken, 2020). Similarly, another approach using *Starcraft II* as a simulation engine used both a scripted course of action and a non-RL-based bot to control the opponent (Goecks, et al., 2021). These approaches typically have custom rewards that encourage certain behaviors and actions, making them dependent on expert knowledge. Furthermore, explicit dependence on scripted or non-learning adversaries can lead to unrealistic assumptions about how a human adversary would act.

In contrast, recent research using self-play has been found to produce realistic human-like behavior for both players in a gaming scenario *without* examples of human play. This was demonstrated by AlphaGo Zero (Silver, et al., 2017), an algorithm leveraging Deep Learning and Monte Carlo Tree Search to achieve human-like play in the game of Go. In the framework they describe, both players start without any Go-specific knowledge beyond the rules of gameplay, building more sophisticated strategies incrementally until they achieve performance at the level of human experts. Unlike other RL approaches, the reward function for this approach is simply +1 for a win, -1 for a loss, and 0 for a draw; no game-specific knowledge is encoded in the reward function. Tree search is a fundamental component of this approach that simulates into the future (planning) to determine what actions have a better expected outcome. More recently, this idea has been combined with "learning from demonstration" (LfD) approaches that seek to add realism to behaviors by learning from observed human behavior (Igl, et al., 2022). Self-play has also been successfully applied to improve video compression size (Mandhane, et al., 2022).

The approach detailed in this work applies self-play, Deep Learning-based Reinforcement Learning, and Monte Carlo Tree Search to electronic warfare (EW). The basic concepts of EW—transmitting and receiving information and jamming opponent communications—are encoded into a simple grid-bound scenario spanning multiple domains and with multiple asset types. Two "AI" (Artificial Intelligence) agents are trained using self-play with only binary feedback (win/loss) as draws are not allowed.

METHOD

SPICE Framework

We introduce the SPICE framework—Strategic Planning Intelligence for Cognitive Electronic Warfare—as a collection of software tools designed to demonstrate the use of AI/ML to achieve human-level constructive behavior in an electronic warfare (EW) environment.

The SPICE architecture three main components: a simulation environment, an intelligent agent engine, and a web-based user interface (UI). By design, the simulation environment is a simple model of an electronic warfare scenario. The SPICE UI is designed with two purposes: to visualize the sequence of states and actions in a scenario and to interact with the agents as a human player. A screenshot is shown in Figure 1. The UI is written in JavaScript using the React framework. Finally, the intelligent agent engine contains a collection of reusable software components for training AI agents and constructing rules-based agents using behavior trees.

Figure 1. Example screenshot from the SPICE UI. The left frame visualizes the environment and allows interaction, while the right frames provide high-level information to a human decision-maker.

The simulation environment was designed to balance two objectives: modeling the essential features of EW while maintaining simplicity. Actors are placed on a small grid and can take actions to move between grid cells. Furthermore, each actor can take certain actions on "bands" representing communications channels. These channels can be obstructed by obstacles and interfered with by an actor taking a jamming action. Information is represented as "intel" points that are transferred between actors when they have a clear communication channel. The scenario is "won" by the player that transmits the most intel points to an actor denoted as the "RX target".

A discrete band and point-based system for was chosen as an abstract method of modeling communications channels (used for communicating intelligence) irrespective of the underlying hardware or software implementation. A "band" might be a literal frequency band *or* an abstract protocol spread across a set of frequencies. Similarly, each "intel" point represents a unit of information and may be an actual measure of data volume (e.g., megabytes) or a less precise unit such as orders. For bands and intelligence, the goal is to model the flow of information across multiple channels with different and time-varying characteristics.

Neural Network-Informed Monte Carlo Tree Search

The field of machine learning (ML) is often divided into three categories: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is focused on finding a mapping from example inputs and outputs that generalizes to unseen examples; Deep Learning (which uses neural networks) is typically applied in a supervised fashion. Unsupervised learning approaches, in contrast, learn the structure and distribution of data *without* the external feedback of a desired output. Finally, the focus in reinforcement learning (RL) is on optimizing a time-dependent scalar (reward) through a sequence of actions.

In its simplest form, the goal in RL is to find a *policy* $\pi(a \mid s)$ that gives a probability of taking an action, a, for a given state, s, that maximizes rewards, $r(s, a)$, when sampled from consecutively. Each action affects the next state, s', for which another action a' is sampled from the policy. Often, this is done through estimating a separate function *value* function $V(s)$ or $Q(s, a)$ that estimates the total future reward received. Deep RL simply uses neural networks to model π and V or Q.

AlphaZero (Silver, et al., 2018) is a recent innovation that augments Monte Carlo Tree Search (MCTS) with principles from Deep Reinforcement Learning. Monte Carlo Tree Search (MCTS) is a well-established technique for producing intelligent behavior in game play. In simple terms, the MCTS algorithm "looks ahead" into the future by simulating a sequence of actions and counteractions for the current player and its opponent. Traditionally, heuristics are employed to estimate the utility of a future state and guide action selection toward strategically useful choices. AlphaZero replaces these with fully differentiable neural networks. A *policy* network outputs a distribution over the action space of a certain state, guiding action selection in place of a heuristic. Similarly, a *value* network estimates the total reward that will be achieved by the agent at the end of the game.

Self-play is employed to update the parameters of the policy and value networks. This approach pits equally matched agents against each other. Initially, these networks are randomly initialized, resulting in agents that take erratic and incoherent actions. The resulting sequences of states and actions—called episodes—are collected and used to optimize the value and policy networks. After iterating this process many times, the value network tends to be a better estimate of the game outcome and the policy network tends toward selecting actions that lead to better outcomes.

Scenarios

As SPICE is a general framework allowing for *many* types of EW scenarios, it is impossible to evaluate performance on all possible scenarios. To this end, two representative scenarios were defined that highlight key AI capabilities of the system.

A simple scenario is shown in Figure 2. This scenario has four actors—two controlled by the "blue" player and two by the "red" player. The goal is for each player to transmit as much intel as possible to their tactical operations center (TOC) in the top corners; the player that transmits the most intel is considered the winner of that episode. There are three bands that the players may transmit over, and they are also allowed to jam each other and prevent the transmission of intel on the jammed band. Jamming occurs when a jamming asset has line-of-sight to another asset on the same band. The decision to limit the scenario to three bands was made to allow for more complex strategies than one or two bands but keep the action space manageable and easy to understand.

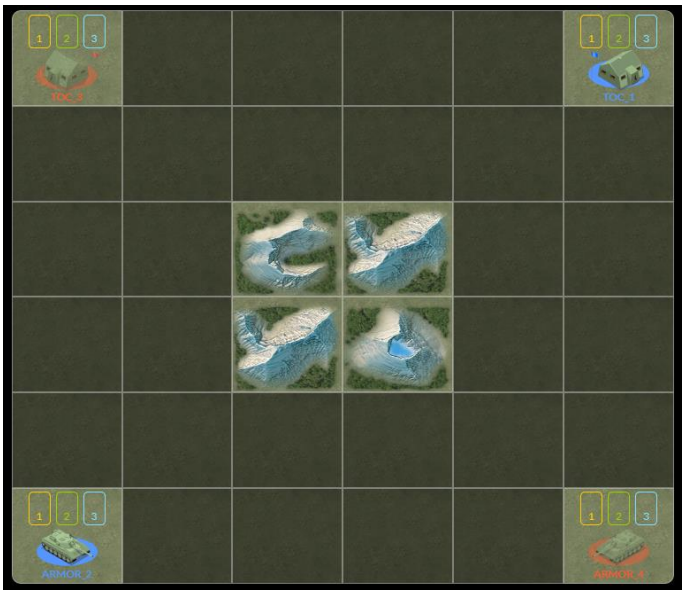

Figure 2. Initial actor positions in simple scenario. The bottom two actors are tanks and the top two are TOCs.

Transmission of intel between actors is only allowed when the following conditions are true:

1. Actors A and B are part of the same force,
2. Actor A has nonzero intel points,
3. Actors A and B have "line-of-sight" (no obstacles obstruct the path of transmission),
4. Actor A is transmitting on band n,
5. Actor B is receiving on band n, and
6. No actors with line-of-sight to B are jamming on band n.

When these conditions are met, transmission occurs at the end of each time step.

Each actor is initialized with 30 points of health. When actors move, their health goes down. Similarly, any enabled band action reduces the health of each actor. When an actor depletes its health, it can no longer take actions.

To prevent ties, a half point advantage is given to the second (red) player at the start of the game. During gameplay, the score is incremented by integer amounts directly corresponding to the amount of intel transmitted to the RX target. However, the value of the score is not directly tied to the reward function optimized by the agent. Instead, the reward is a positive/negative value given by the sign of the difference in scores (denoted by $x_\bullet$):

$$r_p = \text{sign}\left(x_p - x_{p' \neq p}\right), \qquad p, p' \in \{\text{red}, \text{blue}\}, \tag{1}$$

and is only provided to the agent at the end of the episode (the terminal state). Thus, the half point score advantage ensures that the scores of the players are never equal.

A more complex scenario was also defined to highlight multi-agent strategies and is shown in Figure 3. Like the simple scenario, the score is directly tied to the number of intel points transmitted to the target entity (UUV). However, each actor can transmit on a different subset of bands (see Table 1). A simple strategy that transmits the two points of intel from the Plane to the UUV may result in a win but will be easily surpassed by a sophisticated strategy that transmits intel from the TOC to the UUV through the Plane.

Table 1. Allowed EW actions and initial intel points.

		Initial Intel	Band 1	Band 2	Band 3	Range
Actor	UUV (RX target)	0	-	-	RX	-
	Plane	2	TX/RX/Jam	TX/RX/Jam	TX/RX/Jam	2.5
	TOC	5	TX	TX	-	1.5
	Tank	0	TX/RX/Jam	TX/RX/Jam	-	2.0

Figure 3. Initial actor positions in complex scenario.

A baseline agent was defined using behavior trees to simplify comparison. A graphical depiction of the behavior tree is shown in Figure 4. This agent only transmits intel from the Plane to the UUV, limiting the maximum amount of intel transmitted to the UUV at two points.

Training

Three independent agents were trained for 500 iterations using the AlphaZero self-play algorithm. No symmetries were leveraged, allowing the behavior for each player to be fully independent. Each iteration consisted of 100 episodes, and for each step in each episode the MCTS algorithm was iterated 25 times.

At the end of each episode, each experience—a (s, a, r) tuple recording the state s, action a taken, and final reward r at the end of the game—for each player was added to an experience buffer of the last 20 episodes. The value and policy networks for each player were then trained with s as the input and a or r as the target, respectively, using stochastic gradient descent.

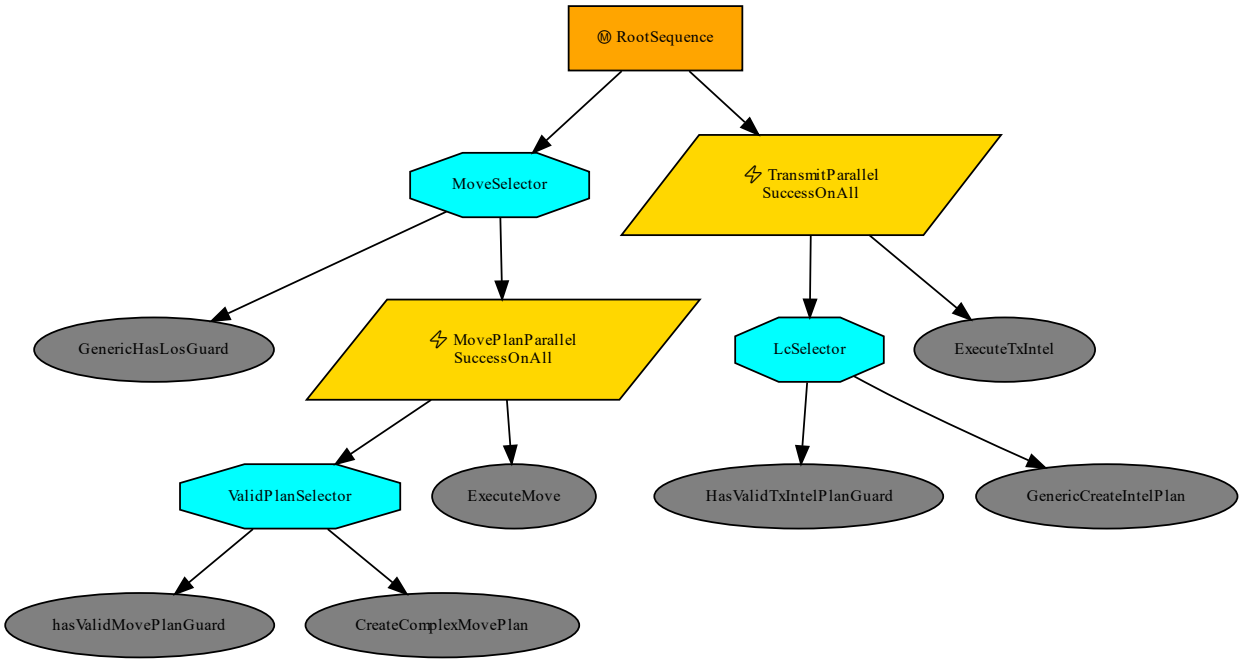

Figure 4. Baseline agent behavior tree.

One important assumption in these scenarios is that the state of the game is fully observable. This implies that there is a bijection between the world state and its representation s; in other words, given two representations, s and s', the assertion that $s = s'$ implies that they represent the same world state. This is an important consideration for Monte Carlo Tree Search.

A Convolutional Neural Network (CNN) structure was chosen to model the value and policy networks (see Figure 5). All weights except the last layer were shared for the policy and value networks of *both* players.

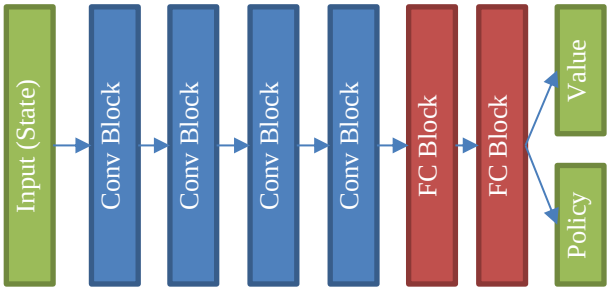

Figure 5. Convolutional Neural Network (CNN) architecture for value and policy networks. Each "Conv Block" consists of a 2D convolution followed by batch normalization and a rectified linear activation. Each "FC Block" contains a fully connected linear layer followed by batch normalization, a rectified linear activation, and dropout.

At the end of each training iteration, the latest model weights are compared to the prior best model weights by pitting MCTS agents against each other using these weights in ten games, five for each friendly/opponent configuration. If the latest weights lead to a win rate greater than 0.6, the weights are "accepted" as the current best. Every other time this occurs, the accepted model is compared against the baseline agent.

Testing

The trained agents are evaluated in two modes: against a consistent baseline agent and against other trained agents. A behavior tree is used by the baseline agent to execute a strategy; see Figure 4 for a graphical depiction. The general algorithm is as follows:

1. Find all grid cells that have line-of-sight to the RX target actor,
2. Select the grid cell with the shortest path from the TX actor, randomly breaking ties,
3. Execute a sequence of actions taking the TX actor to that cell,
4. Pick a band and set up transmission and receiving appropriately.

This algorithm can meet the scenario objectives, despite being relatively weak. In practice, these qualities mean that an AI agent with a low win rate against the baseline agent is likely to have a poor strategy; however, knowing that an agent has a high baseline win rate does not imply that it has a strong strategy. An alternate approach that measures the relative strength of AI agents is referred to here as the "arena performance." This approach pits a collection of agents against each other in a pairwise fashion.

RESULTS

Quantitative Performance

The win rate of the AI agents against the baseline agent over 500 iterations is shown in Figure 6. Initially, all the agents perform extremely poorly against the baseline agent. By the last iteration, however, the agents have an almost perfect win rate against the baseline agent. In other words, the MCTS-based agents have stronger strategies than the decision tree-based agents, despite not being explicitly trained against them.

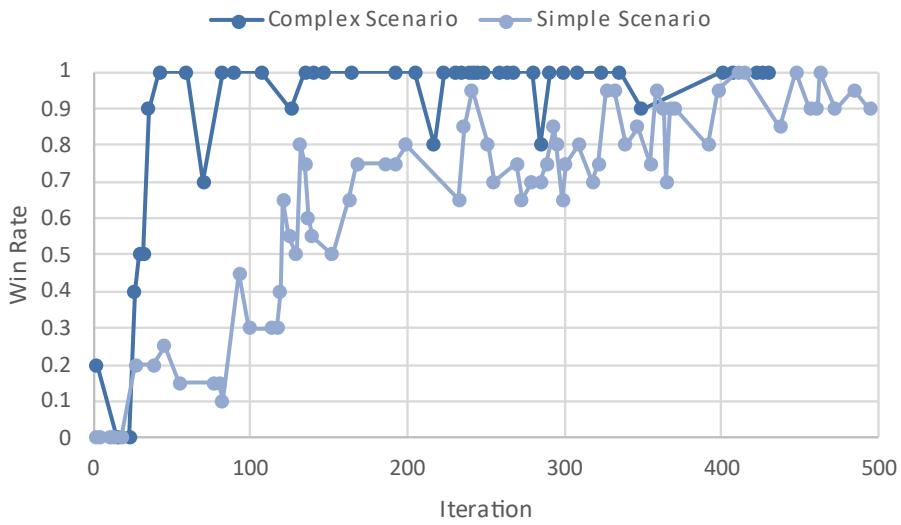

Figure 6. Win rate of AI agents against baseline (decision tree-based) agent, measured across iterations. Agents are only evaluated against the baseline agent for every two accepted models.

The arena performance of the three trained AI agents is shown in Table 2. Ten games were played with each pair of agents and each Red/Blue perspective. From the table, it can be inferred that Agents 1 and 2 are stronger than Agent 3; all three agents are stronger than the baseline agent. Furthermore, Agents 1 and 2 are also stronger as the Red player; this is not surprising, given that the Red player is given a half-point advantage.

Table 2. Arena performance of trained AI agents in the complex scenario. Agent 1 was selected used as the prototypical AI agent for other evaluations. Each cell represents the number of wins of the Blue Player against the Red Player.

		Red Player			
		Agent 1	Agent 2	Agent 3	Baseline
Blue Player	Agent 1	-	0	10	10
	Agent 2	0	-	10	10
	Agent 3	0	0	-	10
	Baseline	0	0	0	-

Qualitative

Despite having no prior knowledge built into them, the AI agents display qualitatively interesting strategies. In the simple scenario, the agents learn a "bait-and-switch" technique. The pattern is as follows:

1. Player sets up actor A to transmit on band n.
2. Player sets up actor B to receive on band $m \neq n$.
3. Opponent sets up actor C to jam on band $k \in \{m, n\}$.
4. Player switches the band used by A or B such that A transmits on band $n' \in \{m, n\}$ and B receives on $m' = n' \neq k$.
5. One point of intel is transmitted from $A \rightarrow B$.
6. Opponent sets up actor C to jam on band $k' = n'$.

In simple terms, the AI agent takes advantage of the fact that the opponent can only jam a single band per step and forces the opponent to choose one of the two bands to transmit, and then switches to the band that the opponent does not transmit on.

In the more complex scenario, the AI agents demonstrate non-trivial behavior by setting up the Plane as a "gateway" between the TOC and the UUV. Again, this requires preliminary actions to set up the gateway: achieving line-of-sight from the TOC to the Plane to the UUV and setting up TX/RX connections on the appropriate bands.

In tandem to displaying constructive behavior, the AI agents also demonstrate behavior that is seemingly irrational or erratic. For instance, in the complex scenario the blue player leads by having the Plane transmit on a band that the UUV cannot receive on. The most likely hypothesis is that moves such as these, while seemingly irrational to a human, have little to no effect on the outcome of the scenario; the reward of provided to the RL algorithm does not consider efficiency. Put concretely, the projected win/loss outcome from taking an unnecessary transmission action is roughly the same as the projected outcome for other actions; the algorithm's preference among these actions can best be attributed to chance. However, it is likely that this result would not occur with a different choice of objective and thus is not a fundamental limit of the approach. For example, a reward proportional to the difference of scores would encourage a "losing" agent to find strategies to lose less severely, in contrast to the win/loss reward used in this scenario that penalizes all degrees of loss equally.

DISCUSSION

A novel application of the AlphaZero algorithm to electronic warfare was presented and analyzed in this work to demonstrate that an AI-based approach can produce human-level strategies in environments given no prior knowledge about friendly or adversarial behavior. Across many domains and fields, a main advantage of an implicit AI-based approach is that it allows a "top-down" perspective for building complex behavior. For planning and decision-making,

this enables a focus on the objectives, physics, and rules of a scenario; if any of these is modified, new agents may simply be trained without modifying the training algorithm itself. In contrast, rules-based methods are often "bottom-up" in the sense that small units of behavior are composed to produce complex behavior. These tend to be highly specific to a scenario and require significant engineering effort to adapt them to changes.

This work demonstrates non-trivial constructive behavior achieved implicitly through self-play and maximization of objectives. This behavior was realized by applying Monte Carlo Tree Search to a discrete simulation environment and using neural network approximations to estimate the scenario outcome and sample actions. Sophisticated strategy was shown even for a simple scenario. Overall, these features make the described AI-based approach well-suited for complex simulation environments where constructive behavior is needed.

Even a seemingly simple scenarios and rules can give rise to complex behavior. This is an important consideration in training environments. Realistic models of adversarial behavior often require significant human effort to develop when using a "bottom-up" approach and are difficult to transfer to new scenarios. AI-based approaches like the one outlined in this work enable a "top-down" approach that can produce non-trivial adversarial behaviors. One limitation is that this AI-discovered behavior is difficult to explain; however, most complex models of behavior have this property, and a "top-down" approach is arguably easier to understand by considering the objectives being maximized.

The SPICE framework is an initial step toward filling the gap for realistic human-like behavior in simulations. This work demonstrates that both novel and effective behavior can be produced through AI/ML; this approach is flexible and general. By applying it to model friendly or adversarial behavior in simulation environments, better models of human behavior—and ultimately more realistic simulations—can be leveraged for training simulations for warfighters and testbeds for new technology or tactics.

ACKNOWLEDGEMENTS

This work was supported General Dynamics Mission Systems through internal research and development funding. The authors would like to recognize the work of Eric Dong, John Rabo, Shari Benko, and Brady Sheehan for their significant contributions to the SPICE framework, laying the foundations for the research detailed in this report.

REFERENCES

Boron, J., & Darken, C. (2020). Developing Combat Behavior through Reinforcement Learning in Wargames and Simulations. *IEEE Conference on Games (CoG)*, (pp. 728-731).

Goecks, V. G., Waytowich, N., Asher, D. E., Park, S. J., Mittrick, M., Richardson, J., . . . Kott, A. (2021). On games and simulators as a platform for development of artificial intelligence for command and control. *The Journal of Defense Modeling and Simulation*, 1-11.

Igl, M., Kim, D., Kuefler, A., Mougin, P., Shah, P., Shiarlis, K., . . . Whiteson, S. (2022). Symphony: Learning Realistic and Diverse. *arXiv*.

Mandhane, A., Zhernov, A., Rauh, M., Gu, C., Wang, M., Xue, F., . . . Mann, T. (2022). MuZero with Self-competition for Rate Control in VP9 Video Compression. *arXiv*, 1-20.

Modeling and Simulation Coordination Office. (2011). *Modeling and Simulation (M&S) Glossary.* Alexandria, VA: Department of Defense.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., . . . Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419).

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., . . . Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354-359.