Strategic maneuver and disruption with reinforcement learning approaches for multi-agent coordination

Derrrik E Asher1, Anjon Basak2, Rolando Fernandez1, Piyush K Sharma1, Erin G Zaroukian1, Christopher D Hsu1, Michael R Dorothy1, Thomas Mahre3, Gerardo Galindo4, Luke Frerichs1, John Rogers1, and John Fossaceca1

Abstract

Reinforcement learning (RL) approaches can illuminate emergent behaviors that facilitate coordination across teams of agents as part of a multi-agent system (MAS), which can provide windows of opportunity in various military tasks. Technologically advancing adversaries pose substantial risks to a friendly nation’s interests and resources. Superior resources alone are not enough to defeat adversaries in modern complex environments because adversaries create standoff in multiple domains against predictable military doctrine-based maneuvers. Therefore, as part of a defense strategy, friendly forces must use strategic maneuvers and disruption to gain superiority in complex multi-faceted domains, such as multi-domain operations (MDOs). One promising avenue for implementing strategic maneuver and disruption to gain superiority over adversaries is through coordination of MAS in future military operations. In this paper, we present overviews of prominent works in the RL domain with their strengths and weaknesses for overcoming the challenges associated with performing autonomous strategic maneuver and disruption in military contexts.

Keywords
Multi-agent systems, reinforcement learning, multi-domain operation, coordination, military scenario, strategic maneuver

1. Introduction

In simple terms, strategic maneuver can be interpreted as a set of agents coordinating their actions to achieve a common goal by overcoming an adversary. Disruption, which is a special case of strategic maneuver, can be represented as the inhibition of an adversary’s coordinated strategic maneuver. Therefore, the use of the terms strategic maneuver and disruption implies that there exists at least two opposing or adversarial sides that are in a dynamic struggle to gain superiority over each other by limiting, inhibiting, or otherwise disrupting their opponent’s coordination or tactics, and imposing their own coordinated tactics.

The nascent surge in the military modernization is motivated by the threat adversaries pose to a friendly nation in multiple domains (e.g., land, sea, air, cyber, electromagnetic, and space),1–3 which threatens national interest in ways beyond conventional warfare. It is expected that future battles will be fought in these complex multi-domain environments,4–6 where artificial intelligence (AI) will guide the tactics, techniques, and procedures (TTPs) of robotic agents working alongside human Soldiers. Such robots will aggregate to form intelligent multi-agent teams coordinating efficiently with human Soldiers to complete the Mission.

The US Combat Capabilities Development Command (DEVCOM) Army Research Laboratory’s (ARL) Essential
Research Programs (ERPs) constitute a specific programmatic path for developing and implementing intelligent multi-agent systems (MASs). Such Army programs provide US defense operations with answers to critical research questions which converge to support the Army Futures Command (AFC) modernization efforts. Artificial Intelligence for Autonomous Maneuver and Mobility (AIMM) and Emerging Overmatch Technologies (EOT) are example ERPs that explicitly focus on enabling the Next-Generation Combat Vehicles (NGCVs) with autonomous sensing, learning, reasoning, planning, and maneuvering capabilities. These future autonomous systems will predict and plan in collaboration with human agents and provide support to the Soldier by autonomously maneuvering in the battlefield. This paper focuses on the autonomous collaboration that needs to take place to have MASs (i.e., human, agent, or human and agent) succeed in future military operations. Integrated and coordinated MAS will require technological advancements focused on collaborative strategic maneuvers beyond our current capabilities to effectively deal with equivalently (peer or near-peer) equipped adversaries. One immediate challenge is to develop teams of agents which can work autonomously and intelligently in a well-coordinated way. This demands that agents observe, orient, decide, and act (OODA-Loop) alongside human Soldiers in mission critical tasks. While novel efforts are being contributed to our understanding of intelligence in multi-agent paradigms, its current interpretation is not well-defined. Recent literature suggests that Reinforcement Learning (RL)-based approaches may provide a viable path toward such technological advances, evidenced by a substantial body of work that will be introduced in this paper.

In this article, we set out to explore algorithmic approaches in the RL domain and their potential application in military environments; specifically on coordination through strategic team maneuver and adversarial coordination disruption in multi-domain operation (MDO). The minimization, limitation, or complete inhibition of coordination in adversarial multi-agent behaviors is a means of exploring and implementing disruption for strategic maneuvers. This avenue of overall reducing the coordination in adversarial forces will be referred to in this article as disruption. Moreover, collaborative strategic maneuvers and disruption can be learned by various RL approaches to inform a defense force of potential avenues for creating windows of opportunity or superiority.

To achieve MAS coordination through strategic maneuver and disruption with RL approaches in simulation environments, we first introduce some of the most prominent RL implementations of recent years. These recent advancements in the RL domain (e.g., AlphaGo) have facilitated the use of more complex multi-agent reinforcement learning (MARL) algorithmic approaches toward eventual real-world application. Furthermore, there have been some frameworks to implement multi-agent coordination in recent years. Together, these efforts may provide a path toward developing and implementing multi-agent coordination for strategic maneuver in multi-robot systems designed for the future battlefield.

In the following sections, (1) the challenges associated with utilizing RL approaches, (2) a taxonomy of RL, (3) a review of recent RL algorithms, and (4) a specific military scenario for application of RL techniques are presented. In general, these RL approaches are shown to align with current research and development programs at ARL.

2. Challenges

Coordination is typically ill-defined in multi-agent tasks and is often used to indicate that a group of agents has performed successfully in some cooperative task domain. In prior work, various novel methods were developed and employed to measure the interdependence between agent actions while performing cooperative tasks, to confirm that these agents had in fact learned to coordinate. The confirmation of coordination is a precursor to establishing that an MAS is capable of working with its partners instead of simply taking actions that result in some measure of optimality. Whereas optimal behavior may be desirable in some circumstances, an agent that is simply acting optimal may result in catastrophic losses on a battlefield if the Mission has changed in some unforeseen way. Therefore, it is critical that MAS for future defense operations have the capability to explicitly coordinate.

For the remainder of this section, some of the challenges associated with developing MAS capable of strategic maneuver and disruption are described, where timescales, capabilities, and local goals may be vastly different (i.e., MDO), but some level of coordination is required. Furthermore, it is assumed that a greater degree of flexible coordination can result in improved (e.g., faster, less losses, non-intuitive strategies, effectively operate with changing capabilities/team composition, etc.) Mission execution.

As an environment changes in response to a dynamic battlefield, the two (at least) adversarial sides may need to reuse plans and predictions to either (1) keep up with or (2) stay ahead of the planning and predictions of an opponent. An RL-trained MAS may be able to learn this dynamic planning and prediction cycle. Alternatively, this can be achieved if the learning agents build an appropriate model of their opponent’s coordinated actions and then take actions to disrupt that coordination.

In an ideal case, an algorithm (or algorithms) selected to guide behavior of an MAS would learn how to deal with changes in the environment, adversary tactics and
capabilities, own capabilities (gain new capabilities or lose previous capabilities), team composition (e.g., swapping of team mates), and mission objectives. However, current state-of-the-art (SOTA) algorithmic approaches are limited by experiences (i.e., MARL and RL approaches). Moreover, team capabilities and composition are typically fixed in most simulations and therefore do not provide sufficient data for an algorithm to operate on. Therefore, in selecting an algorithm to guide the behavior of an MAS intended for producing strategic maneuver and disruption for military defense applications, it is critical that novel events, behaviors, assets, and entities be considered.

In summary, current algorithmic approaches fall short of the required capability in a complex military defense MDO environment. Current shortcomings can be divided into three categories: (1) data requirements, where either data are limited due to the novelty of a situation, a dataset is insufficient to produce accurate predictions, or the data are polluted in some way (e.g., noisy, dirty, or adversarial alteration), (2) limited computational resources, and (3) algorithms do not generalize to situations other than what was encountered during training (e.g., different goals, altered capabilities, or modified team composition) resulting in a narrow or brittle MAS solution.

In this work, we narrow our focus to RL algorithmic approaches that can guide an MAS to overcome the aforementioned challenges for strategic maneuver and disruption in military defense MDO. Technically, RL is a branch of machine learning (ML) algorithms that go beyond building accurate predictions from data, by demonstrating learning with the production of actions in an environment, which can also be considered a form of decision-making, but more accurately described as strategic action selection.

RL agents learn (or are trained) based on a reward function that ultimately determines which is the best action for an agent to select given the current state of that agent (i.e., current situation). For example, an RL agent can interact with an environment to produce experiences that are tied to rewards (based on a reward function), which will result in a learned policy (i.e., a series of state–action pairs). However, we will discuss in later sections that RL algorithms are not yet mature enough to overcome the challenges that would result in human-like adaptability for intelligent decision-making in novel situations or environments. Although RL algorithms have their shortcomings, they appear to be the most promising avenue moving forward toward achieving successful strategic maneuver and disruption in military defense MDO environments.

In the next section, we address RL shortcomings in greater detail to illuminate how overcoming such issues may produce solutions for military defense MDO environments. To do so, an existing taxonomy of RL algorithms is introduced. This effort should provide a better insight into promising RL techniques that may help determine viable avenues for eventual application in US defense MDO environments.

3. RL taxonomy

According to Sutton and Barto, RL is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. This general description of RL indicates that agents select actions in some state space that maximize a reward signal, resulting in a policy, which can be interpreted as a series of state–action pairs (i.e., given a state, an action was learned), and these learned policies determine the trajectory of future actions selected by an agent.

An RL algorithm has one or more of three major components: (1) value function, (2) policy, and (3) model. The value function represents how good it is to be in a state, which can be used implicitly to derive a policy. The policy describes the agent behavior; what actions an agent should take from any given state. The model represents the environment an RL agent interacts with which can be represented as a Markov decision process (MDP). RL algorithms can be taxonomized in three categories based on their use of value functions or policies to guide learning and actions:

- Value-based: value function (implicit policy)
- Policy-based: policy
- Actor Critic: value function and explicit policy

However, the most prominent difference between RL algorithms is whether a model of the environment is utilized (termed model-based or model-free). So, the aforementioned taxonomy can be refined further:

- Model-free RL
  - No environmental model is utilized
  - Value-based and/or policy-based
- Model-based RL
  - Use of an environmental model
  - Value-based and/or policy-based

Finally, combined model-free and model-based RL algorithms fuse together these approaches to get the best of both worlds. In the next two sections, basic RL algorithms are described. Furthermore, technical details are presented for some popular RL algorithms to illuminate their drawbacks, strengths, and potential to be adapted for strategic maneuver and disruption in military domains.

4. Background

4.1. Dynamic programming and MDPs

Dynamic programming (DP) deals with problems that can be solved in a step-wise approach to obtain an optimal
policy through iterations. DP can be used for prediction to compute the value function given a policy. However, DP also assumes that the MDP is known. Therefore, DP can also be used for optimal control: finding the optimal policy by evaluating over several different policies.

There are at least two methods for developing a policy in RL; these are policy iteration and value iteration. In policy iteration, the Bellman expectation equation is used iteratively to evaluate a policy and then an iterative greedy policy improvement method is used to get to an optimal policy. In value iteration, the control problem is solved with the Bellman optimality equation followed by a single optimal policy extraction.

In RL, the terms “model-based” and “model-free” do not refer to the use of a neural network or statistical learning model to predict values, or a next state. Instead, these terms refer strictly to the agent’s use of state transition probability to solve the problem of state-action selection in an environment. An agent can simply use a model (model-based) to predict the next state and the state’s current value. These models of the environment can be provided entirely outside of the learning agent (e.g., by an outside observer that understands the rules of the task). Or the model can be learned by an agent (model-free) through trial-and-error experiences, where the agent in some state selects an action and learns the reward associated with that state–action pair.

If an MDP is unknown, a model-free approach is needed to find a policy. In model-free approaches, an RL agent interacts with the environment to gain experience and learns to take rewarding actions. Furthermore, in model-free prediction, the value function is computed given a policy (i.e., the policy is evaluated). The challenge with a model-free approach is that the value function must be learned with no knowledge of how the world (environment) works.

### 4.2. Monte Carlo temporal difference and Q-learning

Monte Carlo (MC) RL methods solve the prediction problem from direct experiences of an episodic (restarts after some duration) environment. In MC learning, an agent selects actions with or without receiving feedback (i.e., reward signal) until the end of an episode. (An episode is an amount of time (either a preset duration or the episode is stopped because a predetermined condition was met) that an agent has to explore the environment through trial-and-error actions (or state–action pairs).) Once an episode (or batch of episodes) has completed, the value or return for that series of state–action pairs (Each forward step through time (i.e., time step) produces a state–action pair) is computed as the value function. However, temporal difference (TD) learning uses incomplete returns to estimate the value of a state. This is a form of bootstrapping, which provides the agent with an estimate of the predicted reward all the way to the end of an episode, although the agent had only taken one step. In contrast, TD(λ) learning agents can combine MC and TD learning methodologies and take a variable number of steps to estimate the value function.

RL model-free control methods can be divided into on-policy and off-policy learning. On-policy indicates that an agent learns from experiences that are gained through interactions with an environment. This implies that on-policy techniques evaluate a policy by actively following a current policy, then iteratively updating the policy during experience collection. In stark contrast, off-policy methods improve a policy by iterating over previously collected experiences. The experiences that an off-policy method uses to improve the policy can come from non-policy-related behaviors (e.g., demonstrations from an expert in the domain or a series of human-directed state–action pairs). Q-learning is an off-policy learning method that uses a separate, sometimes greedy policy method to sample and estimate the discounted return from the environment while iteratively updating a different policy.

MC methods can yield high variance in the policy (i.e., the agent’s learned behaviors). To solve the issue, an MC approach can be replaced with a TD learning approach to update the Q-value (action-value function) and instead, use an $E$ greedy method to improve the policy. The benefit of TD learning is that an estimated return from the end of an episode can be bootstrapped, which effectively reduces policy variance. An algorithm that uses this particular approach is referred to as SARSA.

SARSA can be modified to accommodate multiple time step intervals so that a policy can be bootstrapped from any point in an episode, which is called n-step SARSA. Furthermore, the time steps can provide variable feedback to the policy with a weighted return across different points, which is referred to as SARSA(λ). Instead of a forward-looking on-policy approach, a backward-looking method can be constructed with eligibility traces for every state–action pair. With this backward-looking method, a Q-value can be updated in proportion to the eligibility trace along with a TD error signal. The combination of these methods provides the SARSA algorithm an opportunity to project forward through episodic time or backwards to generate an RL policy that may be capable of learning in novel situations. However, SARSA is inherently a single-agent algorithm and may need further modifications to be adapted to MAS problem spaces, where partner or adversary actions can change the goals of the environment (i.e., non-stationarity).

### 5. Scalable RL

Scalability of a learning algorithm is one of the major concerns for military tasks in MDO, especially because such
tasks may require a large number of agents to complete an objective. In addition, military tasks can involve multiple sub-tasks each with their own sub-goals, further complicating the scenario. In MDO, it is expected that a sub-goal consists of a myriad of complex strategic maneuvers that would require fast computation for MAS, and an optimal (or at least sufficient) strategy with the use of minimal computational resources (e.g., computing at the tactical edge). Therefore, a scalable RL algorithm must account for (1) environmental and task complexities, and (2) number of agents (partners and adversaries) so each agent can properly select actions as experiences are collected through the RL learning process.

Environmental complexity (i.e., the size of an agent’s state and action spaces) can refer to the number of states available in an environment’s state space along with the number and actions available to agents in that environment. Scalability of an RL algorithm is the capability of computing an optimal policy within reasonable time and computational power for a sufficiently complex state and action space. (Sufficiently, complex is used here as an arbitrary term that is typically identified post hoc through trial-and-error methods (e.g., different state and action spaces)). Environmental complexity also entails the inclusion of additional agents (e.g., expanding to a MAS), where the state space is scaled up to account for the extra agents and the size of the action space is multiplied by that number of agents.

It is not practical to tackle the scalability issue in RL using a table for state–action pairs because continuous domains would make the table untenable, and simultaneously updating the entries of a table for all agents is infeasible within a reasonable amount of time. Even with sufficiently large computational resources (e.g., excessive computer memory) to contain all states, learning across each state–action pair would be too slow. In contrast to utilizing a table for tracking state–action pairs, a solution is to use a non-parametric function approximator (e.g., deep neural network where the weights are the parameters) which approximates values across the entire state space. However, a function approximator must be differentiable, such that a gradient can be calculated to provide the direction of parameter adjustments.

There are two approaches to train value function approximators: (1) incremental methods, and (2) batch methods. Incremental methods use a stochastic gradient to adjust the approximator’s parameters in the direction of the gradient to minimize the error between the estimated and target values. However, the incremental approach is not sample-efficient, and therefore, does not lend itself to scalability. In contrast, batch methods save the data from a set of experiences and use them to compute the error between function approximator estimation and the target value. Batch methods share commonalities with traditional supervised learning, where the outcome is known (e.g., data are labeled) and an error is calculated between the approximator’s estimate value and the actual outcome value. This type of batch learning is typically referred to as experience replay. Repeating this process will lead to a least square error solution. A recent successful example of experience replay was demonstrated with deep Q-networks (DQNs) playing Atari games. Although function approximator methods have shown success in complex environments, it is unlikely that this approach alone will be sufficient to train an MAS for MDO scenarios without accounting for the inclusion of additional agents (i.e., non-stationarity or partial observability).

Compared to value function approximation, policy learning methods rely on Policy Gradient (PG) calculations to explicitly optimize a policy rather than indirectly on a value function. PG has better convergence properties over function approximator methods. (Policy convergence is an important property that implies an algorithmic approach reaches a stable level of reward and performance after a sufficient number of training iterations.) The main reason PG methods are used over value approximation methods is their ability to be effective in high dimensional and continuous action spaces (i.e., scalable in complex environments). In an MC PG (e.g., REINFORCE algorithm), the actual return is multiplied with a score function to get the gradient to adjust the policy parameters to obtain more reward. An MC PG still has high variance and is slow to converge because it uses the entire trajectory of an agent’s state–action pairs across an episode to get the return value. An alternate solution that may surpass the shortcomings of traditional function approximator methods is to use actor-critic approaches.

With actor-critic approaches, a PG equation is modified to use the value function approximation instead of using the true action-value function multiplied by the score (as is done with the REINFORCE algorithm). This indicates that the actor adjusts the policy in the direction that the critic is pointing, so that total cumulative rewards can be maximized. This policy evaluation step by the critic can be done using combined value approximation methods (i.e., MC, TD(0), and TD(λ)). To reduce the variance in the PG, an advantage function can be used. The advantage function tells us how much better one action is over another (Q-value) compared to a general state value function. This implies that the critic must estimate the Q-value. An efficient way to do this is to use TD-error, which is an unbiased sample of the advantage function where the critic approximates one step by the critic can be done using combined value approximation methods (i.e., MC, TD(0), and TD(λ)).
Since MDO involves military tasks where an RL algorithm must have the capability to coordinate with many other agents for optimal strategic maneuvers and disruption, an MAS must be able to scale with a large number of agents and heterogeneous assets. Another important capability of an algorithm is the ability to process the vast observations of complex state spaces (i.e., many agents) and multi-domain environments. In the next sections, we discuss the implication of using different kinds of RL algorithms in MDO for strategic maneuver and disruption.

6. Model-free RL

Model-free algorithms can be divided into off-policy and on-policy algorithms where state–action space can be either continuous or discrete. In this section, the strengths and weaknesses of the model-free algorithms are discussed, along with how they might align with strategic maneuver and disruption, leading to the goals of MDO. The purpose of this analysis is to provide direction toward finding potential algorithmic approaches that may achieve strategic maneuver and disruption in MDO environments.

6.1. DQN

DQN29 is a single RL agent algorithm that was trained to play Atari 2600 games32 where the action space was discrete and the state space was continuous. DQN uses a convolutional neural network trained with Q-learning33 to learn from high-dimensional input (sequential images).

The DQN algorithm is a sample efficient approach because it makes use of all the collected experiences to extract the maximum amount of information possible. DQN is robust enough to be trained using the same hyper-parameters (In ML, a hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are derived via training) to play six different Atari games, where the agent performed better than human experts in three of these games.

However, a drawback with DQN is that there are no theoretical guarantees of a trained neural network achieving a stable Q-Value prediction (i.e., potentially high variance in the trained policies across independent models).

Given that DQN is inherently a single RL agent model, it should be insufficient for strategic maneuver and disruption in MDO. In MDO, multi-agent RL algorithms may be more suitable due to the typical decentralization of the agents during execution time, allowing for agents to operate independent from one another. In addition, the original implementation of DQN only uses a sequence of four observations to learn the Q-value, which is insufficient for strategic maneuver and disruption in MDO. Strategic maneuvers of multiple assets cannot typically be captured within such short time intervals. In fact, this is the primary reason that DQN did not perform well compared to humans in three of the Atari games evaluated (i.e., Q*bert, Seaquest, and Space Invaders). However, there exist some variations of DQN to address this and other weaknesses.

Bootstrap DQN34 is one such variation that learns an ensemble of Q-Networks to improve sample efficiency and overcome the shortcomings of traditional DQN. Action elimination is another method used with DQN to tackle large action spaces.35 DQN with a type of memory (i.e., recurrent neural network) can be used to handle partial observability as well.36 This approach is particularly useful if an agent needs to navigate an environment for task completion. Alternatively, distributional DQN37,38 returns a distribution which can be used to evaluate policy risk and to reduce variance or noise around an optimal solution.

Although DQN and its modified variants are promising for tackling tasks more complicated than simple Atari games, the DQN method inherently lacks a multi-agent prediction mechanism to conduct coordinated tactics, which are required for strategic maneuver and disruption in MDO. Furthermore, DQN is most often too computationally intensive to be used in militarily relevant environments. Finally, DQN algorithmic approaches lack sufficient adaptability for unseen examples (e.g., novel behaviors of partners or entities/obstacles emerge in an environment).

6.2. Deep deterministic policy gradient

In the real world, most regular tasks involve continuous state and action spaces. However, DQN only considers discrete state spaces and low-dimensional action spaces. An alternative approach to DQN, where continuous state and action spaces are handled, is the Deep Deterministic Policy Gradient (DDPG) method. DDPG advances the progress from DQN approaches by combining value function approximation and DPG.39 DDPG uses an actor-critic approach,40 which can overcome the complexities of continuous spaces. This model-free, off-policy, prediction and control algorithm can perform physical control tasks (e.g., cart pole, dexterous manipulation, legged locomotion, or car driving).

Another approach that uses a deep neural network is Trust Region Policy Optimization (TRPO). This method constructs a stochastic policy directly41 without the need for an actor-critic model (not to be confused with an environment model which would make this a model-based approach). Similar to TRPO, guided policy search (GPS)32 is void of an actor-critic model and uses trajectory-guided supervised policy learning along with some additional techniques (e.g., reduction of dimension from visual features, additional information on robot configuration dynamics at the first layer of the network). As a result,
GPS is data-efficient and can be adapted to DDPG if required. PILCO, however, learns a probabilistic model first, then finds an optimal policy. PILCO is highly data-efficient in some problem domains; however, it is computationally demanding. Furthermore, D4PG proposed some improvements over the DDPG algorithm: distribution critic update, distributed parallel actors, N-step returns, and prioritization of the experience replay to achieve a more stable and better solution for different categories of tasks.

From the perspective of strategic maneuvers and disruption, the primary drawback of the DDPG algorithm is that it was designed as a fully decentralized single-agent algorithm (i.e., independent learners). As such, the DDPG algorithm does not facilitate coordination in multi-agent scenarios. Consequently, the resulting strategic maneuvers using DDPG will not result in coordinated team behaviors. Moreover, DDPG is not equipped to handle role-based tasks with multiple objectives which might be a requirement for disruption.

### 6.3 Multi-agent DDPG

RL agent interaction is crucial to AI systems for strategic maneuver and disruption where different agents may need to form teams to effectuate strategic collaboration against an adversary (strategic maneuver) or inhibit the adversary’s coordination (disruption). Q-Learning and PG approaches alone suffer from non-stationarity (Non-stationarity generally refers to an environment where sudden concept/goal/task drift can occur due to dynamic and unknown probability data distribution functions associated with other agents taking actions that change local goals) and high variance, respectively. To overcome these issues, the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm extends an actor-critic approach, which allows it to work for MASs by centralizing agent training. The MADDPG framework adopts a centralized critic and for training and deploys decentralized actors during test time. A critic (one for each agent) receives the policy of every agent, which allows for the development of dependent policies with potentially different reward functions (e.g., MADDPG permits training of adversarial teams with opposing reward functions). Conversely, the actors (i.e., policy networks) only have local knowledge during training and testing. The actor improves the policy iteratively (through training) in a direction consistent with the critic’s evaluation.

Although, a major weakness of MADDPG is that the input to the Q-function increases with the number of agents of the environment (not scalable). This poses a problem for strategic maneuver and disruption in MDO. If agents need to be replaced, added, modified, or removed, retraining may need to take place. In strategic maneuver and disruption, agents may need to switch roles or change capabilities periodically, which poses a major challenge toward adapting MADDPG to military domains. In addition, frequent retraining would make rapid strategic maneuver unlikely. Reducing training time will reduce the computational load on the edge and make rapid strategic maneuver or disruption possible. MADDPG cannot accommodate such extreme cases. For military applications, a robust model of opponents or agents is desired, so that the operational period is maximized (i.e., enough time to execute strategic maneuver and disruption).

A potential modification to MADDPG to address its scalability issues is to form clusters of agents and learn a policy for the clusters instead of each agent individually. In the case of a new event, the need for retraining can be postponed because, in theory, a cluster of agents would have a set of variable capabilities to handle dynamic situations. Furthermore, this would avoid increasing the input space for the Q-function as agents are modified or new agents are introduced. However, the question arises; how can we decompose a task into partially independent sub-tasks with minimum degradation of an optimal group policy?

Although, MADDPG can lead to a set of heterogeneous multi-agent policies capable of diverse tasks, this approach does not scale well beyond a dozen agents. As the number of agents grows, the variance of the PG grows exponentially. Therefore, this approach is not well suited for strategic maneuver and disruption in MDO where more than 40 heterogeneous agents must be accounted for in adversarial contexts. A method for overcoming this scalability issue is the Mean Field Multi-agent RL algorithm, which computes a mean estimation for the neighborhood agents’ Q-value that may result in high error margin when the nearby interaction between agents gets complex. Furthermore, the Evolutionary Population Curriculum (EPC) algorithm was designed to make MADDPG scalable by combining genetic algorithmic approaches with RL. With advances on MADDPG and the successes shown with the approach, it is conceivable that these algorithmic advances could lead to robust demonstrations of strategic maneuvers and disruption within MDO in simulation experiments.

Distinct from MADDPG, the Counterfactual Multi-Agent (COMA) approach uses a single centralized critic for all agents but is designed for discrete action spaces. COMA is more scalable than MADDPG, but it may result in a homogeneous set of policies that could fail with sufficiently different agent capabilities, different local goals, or different reward functions. Similar to MADDPG, Minmax Multi-Agent DDPG (M3DDPG) adds an improvement over the original version of MADDPG, by allowing agents to develop more robust policies against adversaries (i.e., competitive games with opposing reward structures). However, M3DDPG is still unable to handle scenarios when heterogeneous agents are introduced into the system.

Asher et al.
Implementing algorithms into environments with continuous state and action spaces sometimes requires using common techniques to manipulate the inputs or outputs, such as discretizing the state and action spaces or converting the discrete policy output to a continuous output. One example of converting the policy output is the implementation of MADDPG in the OpenAI’s Multi-Agent Particle Environment. In this example, the discrete policy components are used to compute continuous actions. In another perspective, the multi-agent transformer soft double Q-learning algorithm\(^\text{50}\) discretizes the continuous action space into a set of velocity and angular velocity controls which can then be used in a motion model. Although these techniques permit the use of such algorithms in continuous environments, these algorithmic approaches do not train with continuous information, which could limit their efficacy in physical environments for strategic maneuver and disruption.

6.4. Value-based

A recent family of value-based MARL algorithms\(^\text{51}\) has proven to be quite successful in the very complex Starcraft 2 simulation environment\(^\text{52}\) where a centralized joint action-value \(Q_{\text{tot}}\) is learned based on the agents’ local \(Q_a\) values. A decentralized policy is then extracted from the \(Q_a\) by taking the linear argmax operator. This very simple but efficient factorization approach avoids learning the joint action-value which does not scale very well. If new agents are added or agents are replaced with new capabilities, retraining still has to be done. However, it is more scalable compared to MADDPG because the individual Q-values are learned from local observation only which avoid learning the joint-action value by learning a factorized \(Q_{\text{tot}}\). Still the scalability of this family of algorithms can be challenged when there are more than 40 agents. To make it more scalable role-based algorithm, RODE\(^\text{53}\) has been proposed where the agents’ roles are determined by clustering their actions based on their effect on the environment. The algorithm has shown very promising results for large number of agents.

For strategic maneuver and disruption, the RODE algorithm is very promising because groups of agents can be assigned to different roles, where roles can be based on their action and effects on the environment or any other fixed behaviors (for ally or even enemy). The algorithm then can be used for strategic role switching for different groups. Since the action space of different roles are restricted, the algorithm converges very quickly. This algorithm is also fit for strategic use of role-based disruption which we might consider for our future work. Even though RODE is very scalable, it is not clear how we can adapt it for when new agents will be added to the environment because a centralized policy needs to be learned for optimal coordination.

In contrast to the RODE algorithm, a scalable MARL method\(^\text{50}\) deploys an entropy-regularized off-policy method for learning a stochastic value function policy that has been experimentally shown to be able to scale to over 1000 agents. As discussed previously, scalable RL algorithms are concerned with the complexity of the environment—the more agents in the system or team, the larger the state space. RODE is limited as it uses a centralized policy that must be retrained when more agents are introduced into the environment. The algorithm, multi-agent transformer soft double Q-learning, is a centrally trained off-policy learning algorithm, i.e., sharing a central experience replay buffer, with decentralized execution, i.e., each agent makes its own control decisions based on its local observations, not from a central controller. Due to this decentralization scheme, when agents are added or removed from the system, the team is unaffected and continues to execute their policy.

With respect to scalability, training a large MAS (i.e., many agents in the team) is difficult, and it has been shown that even the SOTA algorithms fail to learn performant policies for complex MARL tasks. Multi-agent transformer soft double Q-learning alleviates this scalability issue by using a heuristic during training that allows for the policy to be trained on a smaller set of agents (e.g., four agents tracking four targets in a target-tracking scenario), and the policy has been shown to work with many more agents in execution without any adaptation needed (i.e., tested and evaluated with 1000 agents). The heuristic used during training and execution allows the algorithm to address a dramatic distribution shift in the number of agents: it essentially scales down the large complex observation space at test time into something that is close to what the agent policy was originally trained for. In the military perspective, this formulation is ideal for strategic maneuver and disruption as agents in the field might be lost or gained in situ and might have to account for additional strategic information. A flexible and scalable algorithm provides the capabilities needed to be robust in MDO.

7. Model-based RL

In contrast to model-free RL approaches where agents learn a policy from a stream of experiences (i.e., no prior environmental knowledge), model-based RL approaches enable agents with an a priori functional representation of an environment (i.e., given some pre-designed model of how an environment functions). Model-based RL has a strong advantage over model-free RL by being sample-efficient (i.e., less computational resources needed). This indicates that an agent needs very few samples to learn
how to achieve a goal within some environment. Once the model and the cost function are known, an agent can operate optimally without further sampling (i.e., no additional real-world experiences are needed but simulated experiences are used). However, providing an appropriate environmental model can be quite difficult and in the case of strategic maneuver and disruption within MDO, perhaps impossible.

In model-based RL, a model can be learned via supervised learning. The rules of a situation can provide sufficient information for a model. Also, the model-based approach provides a clear opportunity to reason about the familiar and unfamiliar states of an environment to explore and exploit more effectively. If an environmental model is not accurate, uncertainty can be reasoned about through exploration, similar to a model-free approach. A model-based approach for strategic maneuver and disruption in MDO can be promising when small factors can affect the value and policy drastically.

Potential applications for model-based approaches for strategic maneuver and disruption in MDO can be explored with game-based simulations (e.g., adversarial scenarios involving turret reconnaissance, predator–prey pursuit, target acquisition, or multi-asset fire-fight games with well-structured rules and win conditions) where the state space is relatively small. In game-like cases, learning an environmental model is potentially efficacious, whereas real-world environments may have untenable environmental models. An example in literature is the AlphaGo algorithm, where agents were given a model of the game environment (the game of Go), and the model-based approach performed better than model-free RL approaches. Another disadvantage of model-based approaches is that there are two sources of approximation error: policy (model-free) along with the environmental model (model-based).

Forward search is another example of model-based RL that focuses primarily on planning. Forward search does not solve the entire MDP. It focuses on a sub-MDP space that is simply taken from an agent’s current state. The sub-MDP is solved by forwarding search approaches that select the best action via lookahead techniques. In general, forward search algorithms construct a search tree from the current state as a root, using a model of the environment to lookahead to the expected reward for a series of actions toward the goal.

Simulation-based search is a forward search technique that uses sample-based planning. The samples are generated using a model of the MDP to estimate what might happen in the future. Since the samples are generated based on a model of the environment, the samples reflect the most probable experiences rather than focusing on extraneous, unnecessary, or possibly pointless experiences through environmental exploration (as is done in model-free approaches). This approach might be useful for situation when a strategic maneuver and disruption policy need to be learned within a very short amount of time.

An alternative to forward search methods in model-based RL, MC, or TD techniques can be used in conjunction with a model of an environment. Monte Carlo Tree Search (MCTS) is a specific example of a simulation-based search technique that has proven to be successful in achieving human-like performance in games such as Go. However, if we consider strategic maneuver and disruption in MDO, MCTS needs to be extended to MAS. In fact, the multi-agent version of MCTS has shown promising performance in finding coordination strategies compared to other similar algorithms in grid-based worlds (i.e., discrete state spaces). Furthermore, MCTS can operate in continuous action spaces.

MC search methods can be made more efficient with bootstrapping techniques, such as TD learning (one step, n-step, or TD(λ)), which is referred to as TD search. Alternatively, a function approximator may be used to estimate Q-values instead of either MC or TD approaches.

Providing an accurate enough model for agents to explore viable behaviors that lead to successful outcomes is the core challenge using model-based RL for strategic maneuver and disruption in MDO. Among all the model-based RL techniques explored here, it appears that simulation-based approaches are the most promising because they have the potential to provide robust action selection in situations where agents experience drastic changes to the environment are unseen or novel situations. Since simulation-based search solves an MDP only from the current state, this can be highly promising for minimizing execution time while maintaining a robust policy in dynamic situations, such as strategic maneuver and disruption in MDO.

8. Model-free and model-based RL

Dyna architecture is an example of an algorithm that combines model-free and model-based approaches. In this architecture, gathered experiences are used to learn a model of the environment, where both model-based errors and model-free (gathered) experiences are combined to learn the value function and policy.

For strategic maneuver and disruption in MDO, the Dyna architecture can provide a balance between long-term and short-term needs associated with different tasks. However, in an adversarial scenario (i.e., agents have opposing reward structures), where a value function can vary drastically, model-based approaches might be more beneficial than model-free approaches due to opposing sides adopting new exploitative tactics. Dyna-Q improves on the Dyna-Q algorithm by providing
reward explicitly for exploration (i.e., promoting model-free methods). Although, Dyna-Q+ has been explored minimally in multi-agent paradigms. However, Dyna-Q+ has potential for strategic maneuver and disruption in MDO because of its promising capability of handling unseen examples in policy formation.

9. MAS for strategic maneuver and disruption in MDO

In this section, an adversarial engagement scenario is provided that is centered on the use of selected long-range assets that inherently disrupt friendly force engagement. A legend is shown in Figure 1 to describe the military symbology for selected assets and forces associated with the described MDO scenario. According to MDO59 doctrine, in an armed conflict, adversarial long-range anti-access and area-denial (A2AD) fire systems (e.g., short-range ballistic missiles, long-range surface to air missiles and multiple launch rocket systems, which are protected from aerial attack and hidden in unknown locations) can be used to deny friendly forces freedom of maneuver in a theater of operations. This is accomplished by combining intelligence, surveillance, and reconnaissance (ISR) assets with both lethal and non-lethal fires to attack friendly command structures, sustainment capabilities, and troop formations in the strategic and operational support areas (see Figure 1, OPFOR). These areas are the traditional staging ground for assets (e.g., troops and equipment) operating in the Close Area (see Figure 2). The adversary’s ability to identify and engage targets well behind friendly lines (see Figure 2, Operational Support Area) causes those entities to be geographically separated from the Tactical Support and Close Areas, which effectively raises the attrition rate of friendly forces, referred to as stand-off. Because the forward force is separated from strategic and operational maneuver support, adversarial forces are able to take advantage of this friendly force isolation and destroy them.

The MDO59 doctrine lays down a plan to defeat adversarial A2AD capability (i.e., stand-off), so that strategic and operational maneuvers can enable forward-deployed friendly forces to engage the adversary on favorable terms (i.e., penetrate and disintegrate A2AD systems to exploit freedom of maneuver). Here, we focus only on the penetration and disintegration portions of an engagement (There are six joint functions from the doctrinal phases of a joint operation. These functions are: Command and Control (C2), Intelligence, Fires, Movement and Maneuver, Protection, and Sustainment. Penetration and disintegration are part of the “Fires” and “Movement and Maneuver” joint functions) with adversarial A2AD systems by friendly (BLUEFOR) field army and corps that may entail the use of autonomous MAS in future battles. Furthermore, it is speculated that all of the symbols shown in Figure 1 for both friendly (BLUEFOR) and adversary (OPFOR) forces will contain autonomy-enabled formations (e.g., Robotic Combat Vehicles (RCV), automatic targeting systems, ground and aerial robotic ISR assets). Scenario diagrams for strategic maneuver and disruption using this symbology with the autonomy-enabled formations are shown, respectively, in Figures 2 and 3.

The adversarial A2AD fire systems create stand-off by attacking the strategic and operational support areas as shown in Figure 2. The friendly fires and air defense forces receive targeted intelligence from space and high-altitude surveillance (not shown) to strike high-value targets (i.e., MLRS) within narrow time windows to reduce adversarial
position adjustments. In addition to surveillance, strategic stimulation-see-strike can be employed to penetrate and disintegrate adversarial long-range fire systems.\textsuperscript{59}

MARL can be used to strategically illuminate and track the locations of adversarial targets by exploiting adversarial doctrine and local observations from adversarial actions. Furthermore, MARL trained autonomy-enabled formations with a combination of highly mobile and distributed air and ground fires can begin to overwhelm adversarial long-range air defenses. Friendly forces may utilize MARL approaches trained to exploit adversarial TTPs for strategic maneuver of air defense and ground fires. These autonomy-enabled formations choose geographic locations based on surveillance data collected from strategic air-based stimulation. As adversarial long-range fire systems are neutralized, strategic and operational support units are able to advance (maneuver) toward the forward OPFOR (see Figure 2).

Adversarial forces identify friendly assets in the Operational Support Area using ISR assets and engage friendly forces with long-range fires systems (i.e., MLRS) from the Operational Deep Fires Area. These hostile fires disrupt the friendly force’s ability to conduct traditional support operations in that area, which in turn, causes such activities to take place further back from the forward line of troops (FLOT). This creates geographical stand-off by extending the battlefield and straining supply lines. Furthermore, this permits the hostile maneuver forces to engage friendly forces in the Close Area on terms favorable to the enemy fait accompli. (fait accompli is a term that is used here to describe an event has already been decided before those affected hear about it, leaving them with no option but to accept it.) According to MDO doctrine, to eliminate stand-off, friendly artillery systems must identify, engage, and destroy hostile fires, and ISR assets before they can be deployed. Friendly Special Operations Forces (SOF) assist this effort by disrupting supply and command and control (C2) nodes and providing targeting data for Joint Fires. This creates gaps in the enemy A2AD umbrella which can be exploited by maneuver commanders. Under this coverage, friendly maneuver penetrates and then exploits gaps in the Close and Deep Maneuver Areas.

Strategic formations of joint forces in the Close and Deep Areas, starting from the Operational Area, may be autonomous-enabled formations (i.e., MAS) utilizing MARL trained policies to exploit adversarial TTPs (from doctrine), local observations, and ISR-gathered information. Joint forces will coordinate between their ISR and long-range precision fire capabilities to provide support for the forward-deployed BLUEFOR forces as shown in

\textbf{Figure 2.} Adversarial forces (OPFOR) use long-range missile and rocket fire to disrupt or destroy sustainment operations in the friendly (BLUEFOR) Strategic Support Area, which prevents friendly forces from engaging enemy maneuver elements in the Close Area on favorable terms. To counter this strategy, BLUEFOR conducts counterfire missions to destroy OPFOR long-range fire systems located in the Deep Fires Area (blue arrow; Target No. FT0002). The three-pronged arrow emanating from the BLUEFOR SOF in the Deep Maneuver Area represents a “Disrupt” tactic which breaks up the adversary’s formation and tempo.\textsuperscript{60}
Figure 2. With support from strategic and operational units, forward forces with autonomous-enabled formations can coordinate in Close and Deep Areas to isolate and defeat adversarial assets. This leads to the elimination of the adversarial forward maneuvering forces, (OPFOR) leaves long-range fire systems vulnerable to ground attack (disintegration) in Figure 2.

Joint Fires (i.e., friendly forces or BLUEFOR) suppress or neutralize adversarial long-range fire systems, allowing friendly Maneuver Forces to move-in and defeat OPFOR in the Close Area (see Figure 3). Friendly Maneuver Forces then exploit this advantage by destroying adversarial enablers in the Deep Maneuver Area (see D in Figure 3). This causes the remaining adversarial maneuver formations to withdraw from the Close Area and establish a new front in the Deep Maneuver Area. This process repeats until strategic objectives are met or OPFOR is defeated. These coordinated activities could in theory be achieved in a collaboration between human Soldiers and an autonomous MAS. Furthermore, given that there is active research in the development and deployment of such autonomous systems, it is expected that battlefields of the future will need to consider scenarios like this for planning of strategic maneuver and disruption.

This section has provided a scenario where autonomous-enabled formations may be trained with MARL approaches; however, the specific RL approaches to perform in such complex MDO environments have not been tested or may not yet exist.

10. Insights and conclusion

Adversarial actors are becoming more advanced due to a number of factors, including scientific and technological progress made through proxy conflicts that test novel technologies. Coordinated strategic maneuver and disruption can be used by defense forces to give certain advantages over an adversary in future MAS autonomous warfare. In this article, some of the most prominent RL algorithms were discussed to uncover viable candidates for training MAS that can effectively perform strategic maneuver and disruption toward opening windows of opportunity in potential future military operations. A taxonomy of RL approaches was described with an overview for the most prominent RL algorithms.

It was found that most RL algorithms lack the capability of handling the complexities associated with potential future conflicts due to differences in training and test factors. However, two approaches appeared to utilize a combination of techniques (i.e., Dyna architecture and simulation-based search), which possess the potential to overcome both the inherent complexities associated with
strategic maneuver and disruption in MDO, and the ever-changing dynamics of a battlefield scenario.

Since military engagement scenarios are highly dynamic and complex, it is no surprise that all current RL techniques fail to meet one or more challenges associated with these scenarios and are not mature enough to be deployed off the shelf. A table is provided to summarize the capabilities and limitations of the RL algorithms described in this paper (see Table 1). It is important to note that none of these approaches provide guarantees on performance when the testing environment is different in any way from the training environment, resulting in brittle or narrow solutions in the military engagement domain. Specifically, agents will typically need to be retrained if any changes are applied to the state or actions spaces after training has completed (e.g., adversary behavior or capabilities, own capabilities, team composition, strategic team positioning, or mission objectives). Furthermore, it is difficult to conceive of an RL technique that can handle the common concept of reinforcements to the battle if not explicitly trained for their inclusion. In addition, no RL technique has verified the ability to function properly for much longer durations than trained for (e.g., trained for a 10-h engagement but must operate for days or weeks in deployment to overcome an adversary in practice).

Finally, the algorithms described in this paper are not suited to perform multiple or changing objectives, which is critical for strategic maneuver and disruption in military engagement scenarios.

There are three algorithms (EPC, MAMCTS, and Dyna-Q/Q+) that stand out as potentially viable options for training an MAS for strategic maneuver and disruption in MDO. Although these algorithms stand out, they all have their drawbacks that would cause them to fail under certain conditions in military engagement scenarios.

Among the list of algorithms shown in Table 1, EPC is one with the potential to perform decently in some portions of the military MDO scenario shown in Figures 2 and 3. EPC uses an individual reward, can be scaled up to 40 agents (indicating some level of handling reinforcements or degraded forces), and operates in a continuous space (desired for small-scale rapid actions), which implies it can operate in tactical level engagements where a variable number of heterogeneous agents must coordinate actions to defeat a dynamic adversary. However, given that EPC is model-free, the state and action spaces may be quite large, making training potentially untenable for complex or multi-domain tasks.

In contrast to EPC, MAMCTS has been shown to scale up to 1000 agents (after training), operates with individual rewards as well (indicating support for heterogeneous agents and different objectives), functions in both continuous and discrete environments (allows for both tactical and operational or strategic level engagements), but is strictly model-based (requires a model of the environment which may be unavailable), which may lead to substantial difficulties for practical implementation in military engagement scenarios.

Although Dyna-Q/Q+ has been unverified in scalability (unlike EPC and MAMCTS), is inherently a single-agent algorithm (indicating the need for multiple algorithmic instantiations for multi-agent implementation), and may not support variations in agents (to accommodate reinforcements or lost forces), it can operate in continuous and discrete domains (for both tactical and operational or strategic level engagements), and most importantly, utilizes both model-free and model-based approaches, which allows for a basic model of the environment to facilitate exploration, unlike the drawbacks of strictly model-free or model-based approaches alone. The strength of Dyna over EPC and MAMCTS is that it can expand on a simple model of the environment (model-based) with state and action space exploration (model-free) to encounter novel tactics and strategies from simple known doctrine-based maneuver. Furthermore, a multi-agent extension of Dyna could permit this technique to match or overcome the strengths of both EPC and MAMCTS, which might be achieved with a centralized training signal (as is done with the majority of MARL algorithms). Therefore, the combination of model-free and model-based approaches gives algorithms such as Dyna, an advantage over other algorithms for application in coordinated MAS military engagement scenarios. As a specific example, if we provide an algorithm a model (model-based) of how to engage an adversary (e.g., maneuver formation for tactical engagement), and then allow the algorithm to explore the state and action spaces (model-free), then the result may be a more effective engagement approach that minimizes friendly force losses and maximizes adversarial casualties.

In general, for a military engagement scenario, we need an RL algorithm that is robust: can maintain the training performance with changes in the testing environment and does not need to be retrained if new capabilities are added, scalable: capable of training a large number of heterogeneous agents which can be reinforced during testing and capable of performing for a long operational duration. We think the future direction of the military should be focused on integrating the strengths of model-free and model-based algorithms for coordinated MAS to execute strategic maneuver and disruption. Alternatively, if we cannot use model-based algorithms (due to the lack of a model), there needs to be a focus on scalable model-free algorithms, where centralized training approaches (i.e., centralized training with decentralized execution—CTDE) can be a computational bottleneck but essential for optimality (i.e., best performance given a reward structure). To remove the bottleneck of centralized training, finding partially independent sub-tasks in the state space and a hybrid training approach.
Table 1. Summary for findings from algorithmic review of recent literature.

| Algorithm          | Algorithm type | Action-space type | Model type     | Scalable to number of agents | Change number of agents after training | Multi-agent reward signal | Algorithmic requirement for heterogeneous agents |
|--------------------|----------------|-------------------|----------------|------------------------------|------------------------------------------|---------------------------|-----------------------------------------------|
| PG                 | Single agent   | Continuous        | Model-free     | Unverified                   | No                                       | N/A                       | Multiple instantiations of the algorithm      |
| DQN                | Single agent   | Discrete          | Model-free     | Unverified                   | No                                       | N/A                       | Multiple instantiations of the algorithm      |
| TRPO               | Single agent   | Continuous        | Model-free     | Unverified                   | No                                       | N/A                       | Multiple instantiations of the algorithm      |
| DDPG               | Single agent   | Continuous        | Model-free     | Unverified                   | No                                       | N/A                       | Multiple instantiations of the algorithm      |
| MADDPG             | Multi-agent    | Continuous        | Model-free     | 10                           | No                                       | Individual                | Single instantiation of the algorithm         |
| Mean Field Q-learning(MF-Q) | Multi-agent    | Continuous        | Model-free     | 10                           | No                                       | Global                    | Single instantiation of the algorithm         |
| M3DDPG             | Multi-agent    | Continuous        | Model-free     | 10                           | No                                       | Individual                | Single instantiation of the algorithm         |
| EPC                | Multi-agent    | Continuous        | Model-free     | 40                           | Yes                                      | Individual                | Single instantiation of the algorithm         |
| QMIX               | Multi-agent    | Continuous        | Model-free     | 8                            | No                                       | Global                    | Single instantiation of the algorithm         |
| Roles to Decompose(RODE) | Multi-agent   | Continuous        | Model-free     | 27                           | No                                       | Global                    | Single instantiation of the algorithm         |
| Multi-agent transformer soft double Q-learning | Multi-agent | Continuous and continuous | Model-based | Unverified                   | No                                       | N/A                       | Multiple instantiations of the algorithm      |
| MCTS               | Single agent   | Discrete and continuous | Model-based | Unverified                   | No                                       | N/A                       | Multiple instantiations of the algorithm      |
| MAMCTS             | Multi-agent    | Discrete and continuous | Model-based | 1000                         | Yes                                      | Individual                | Single instantiation of the algorithm         |
| Dyna-Q/Q+          | Single agent   | Discrete and continuous | Both           | Unverified                   | No                                       | N/A                       | Multiple instantiations of the algorithm      |

PG: Policy Gradient; DQN: deep Q-networks; TRPO: Trust region policy optimization; DDPG: Deep Deterministic Policy Gradient; MADDPG: Multi-Agent Deep Deterministic Policy Gradient; M3DDPG: Minmax Multi-Agent Deep Deterministic Policy Gradient; EPC: Evolutionary Population Curriculum; MCTS: Monte Carlo Tree Search; MAMCTS: multi-agent version of MCTS.
(between centralized and decentralized) might be an alternate potential future direction for model-free techniques.

Furthermore, additional investigation is needed to illuminate military strategy that facilitates the utilization of MAS in engagement scenarios. In obvious cases, it is desirable to send fully autonomous MAS into high-risk situations (i.e., where expected causality rates are high); however, it is insufficient to simply expect that an MAS will be capable of achieving the Mission in the absence of human oversight or intervention due to current technological limitations. Therefore, in future work, it is expected that a robust set of engagement scenarios will be identified. Finally, this line of work will lead to the eventual integration of semi/fully-autonomous MAS for coordinated strategic maneuver and disruption where possible.

Acknowledgment
This project was supported in part by an appointment to the Science Education Programs at National Institutes of Health (NIH), administered by ORAU through the US Department of Energy Oak Ridge Institute for Science and Education. Furthermore, this work was supported by the US DEVCOM ARL ERPs, which have provided the resources to describe and pursue solutions toward operationalizing science for strategic maneuver and disruption with MAS in MDO. ARL’s ERPs provide a programmatic path for developing and implementing intelligent MAS. Given that Army research programs provide US defense operations with answers to critical research questions, these ERPs specifically enable research that can provide a path toward coordinated autonomous MAS that can overcome the challenges associated with, the complexity of an environment, (2) adversary tactics and capabilities, (3) own capabilities (i.e., gain new capabilities, lose previous capabilities, or have capabilities altered), (4) team composition (e.g., adding, removing, or swapping of team mates), (5) strategic team positioning, entry, and navigate (maneuver) to support forces and overwhelm an adversary, and (6) mission objectives.

Funding
The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iDs
Derrik E Asher https://orcid.org/0000-0002-1217-8581
Anjon Basak https://orcid.org/0000-0001-8427-038X

References
1. Sharma PK, Dennison M and Raglin A. Iot solutions with multi-sensor fusion and signal-image encoding for secure data transfer and decision making, 2020, http://www.ibai-publishing.org/html/proceedings_2020/pdf/proceedings_book_MDA-AI&PR_2020.pdf
2. Sharma PK and Raglin A. Image-audio encoding to improve C2 decision-making in multi-domain environment. In: 25th International Command and Control Research and Technology Symposium, Southampton, UK, November 2020.
3. Sharma PK. Heterogeneous noisy short signal camouflage in multi-domain environment decision-making. Trans Mass Data Anal Image Sign 2020; 11: 3–26, http://www.ibai-publishing.org/journal/issue_massdata/2020_september/massdata_11_1_3_26.php
4. Sharma PK and Raglin A. Image-audio encoding for information camouflage and improving malware pattern analysis. In: 2018 17th IEEE international conference on machine learning and applications (ICMLA), Orlando, FL, 17–20 December 2018, pp. 1059–1064. New York: IEEE.
5. Sharma PK and Raglin A. Iot in smart cities: Exploring information theoretic and deep learning models to improve parking solutions. In: IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), August 2019, Leicester, UK. pp. 1860–1865.
6. Sharma PK and Raglin A. Iot: smart city parking solutions with metric-Chisini-Jensen-Shannon divergence based kernels. In: The 5th IEEE international conference on internet of people (IoP), Leicester, UK, August, 2019.
7. Autonomous Maneuver and Mobility, https://www.arl.army.mil/what-we-do/essential-research-programs/
8. Sharma PK, Zaroukian E, Fernandez R, et al. Survey of recent multi-agent reinforcement learning algorithms utilizing centralized training. In: Artificial intelligence and machine learning in multi-domain operations applications III, 2021, vol. 11746, p. 117462K. Bellingham, WA: The International Society for Optics and Photonics. DOI: 10.1117/12.2585808.
9. Silver D, Huang A, Maddison CI, et al. Mastering the game of go with deep neural networks and tree search. Nature 2016; 529: 484–489.
10. Pierpaoli P, Ravichandar H, Waytowich N, et al. Inferring and learning multi-robot policies by observing an expert. arXiv preprint arXiv:190907887 2019.
11. Barton SL and Asher D. Reinforcement learning framework for collaborative agents interacting with soldiers in dynamic military contexts. In: Next-generation analyst VI, 2018, vol. 10653, p. 1065303. Bellingham, WA: The International Society for Optics and Photonics. DOI: 10.1117/12.2303827.
12. Barton SL, Waytowich NR and Asher DE. Coordination-driven learning in multi-agent problem spaces. arXiv preprint arXiv:180904918 2018.
13. Caylor JP, Barton SL, Zaroukian EG, et al. Classification of military occupational specialty codes for agent learning in human-agent teams. In: Artificial intelligence and machine learning for multi-domain operations applications, 10 May 2019, vol.
11006, p. 110060W. Bellingham, WA: The International Society for Optics and Photonics. DOI: 10.1117/12.2518473.

15. Rodriguez SS, Chen J, Deep H, et al. Measuring complacency in humans interacting with autonomous agents in a multi-agent system. In: Artificial intelligence and machine learning for multi-domain operations applications II, 21 April 2020, vol. 11413, p. 114130U. Bellingham, WA: The International Society for Optics and Photonics. DOI: 10.1117/12.2559474.

16. Asher D, Garber-Barron M, Rodriguez S, et al. Multi-agent coordination profiles through state space perturbations. In: 2019 international conference on computational science and computational intelligence (CSCI), Las Vegas, NV, 5–7 December 2019, pp. 249–252. New York: IEEE.

17. Asher D, Barton S, Zaroukian E, et al. Effect of cooperative team size on coordination in adaptive multi-agent systems. In: Artificial intelligence and machine learning for multi-domain operations applications, 10 May 2019, vol. 11006, p. 110060Z. Bellingham, WA: The International Society for Optics and Photonics. DOI: 10.1117/12.2518626.

18. Asher DE, Zaroukian E, Perelman B, et al. Multi-agent collaboration with ergodic spatial distributions. In: Artificial intelligence and machine learning for multi-domain operations applications II, 21 April 2020, vol. 11413, p. 114131N. Bellingham, WA: The International Society for Optics and Photonics.

19. Barton S, Waytowich N, Zaroukian E, et al. Measuring collaborative emergent behavior in multi-agent reinforcement learning. In: International conference on human systems engineering and design: future trends and applications, 16–18 September 2019, pp. 422–427. Germany: Springer.

20. Fernandez R, Basak A, Howell B, et al. Multi-agent collaboration in an adversarial turret reconnaissance task. In: International conference on intelligent human systems integration, 2021, pp. 38–43. Springer. DOI: 10.1007/978-3-030-68017-6_6.

21. Fernandez R, Zaroukian E, Humann JD, et al. Emergent heterogeneous strategies from homogeneous capabilities in multi-agent systems. Springer, 2021. pp. 491–498.

22. Zaroukian E, Rodriguez SS, Barton SL, et al. Algorithmically identifying strategies in multi-agent game-theoretic environments. In: Artificial intelligence and machine learning for multi-domain operations applications, 2019, vol. 11006, p. 1100614. Bellingham, WA: The International Society for Optics and Photonics. DOI: 10.1117/12.2518609.

23. Zaroukian E, Basak A, Sharma PK, et al. Emergent reinforcement learning behaviors through novel testing conditions. In: Artificial intelligence and machine learning for multi-domain operations applications III, April 2021, vol. 11746, p. 117460T. Bellingham, WA: The International Society for Optics and Photonics. DOI: 10.1117/12.2585627.

24. Nguyen TT, Nguyen ND and Nahavandi S. Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications. IEEE Trans Cybern 2020; 50: 3826–3839.

25. Dulac-Arnold G, Mankowitz D and Hester T. Challenges of real-world reinforcement learning. arXiv preprint arXiv:190412901v1 2019.

26. Sutton RS and Barto AG. Reinforcement learning: an introduction. 2nd ed. Cambridge, MA: The MIT Press, 2018, http://incompleteideas.net/book/the-book-2nd.html.

27. Howard RA. Dynamic programming and Markov processes. New York: John Wiley; 1960.

28. Hammersley J. Monte Carlo methods. Berlin: Springer Science & Business Media; 2013.

29. Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 2013.

30. Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learning 1992; 8: 229–256.

31. Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In: Balcan MF and Weinberger KQ (eds.), Proceedings of the 33rd International Conference on Machine Learning, Proceedings of Machine Learning Research, 2016, Vol. 48. New York, USA: PMLR, pp. 1928–1937.

32. Bellemare MG, Naddaf Y, Veness J, et al. The arcade learning environment: an evaluation platform for general agents. J Artif Intell Res 2013; 47: 253–279.

33. Watkins CJ and Dayan P. Q-learning. Mach Learning 1992; 8: 279–292.

34. Osband I, Blundell C, Pritzel A, et al. Deep exploration via bootstrapped DQN. In: Advances in neural information processing systems, 2016, pp. 4026–4034, https://proceedings.neurips.cc/paper/2016/file/8d8818c8e140c64c743113f563cf750f-Paper.pdf

35. Zahavy T, Haroush M, Merlis N, et al. Learn what not to learn: action elimination with deep reinforcement learning. In: Advances in neural information processing systems, pp. 3562–3573. DOI: 10.48550/arXiv.1809.02121.

36. Hausknecht M and Stone P. Deep recurrent Q-learning for partially observable MDPs. In: 2015 AAAI fall symposium series, 2015, https://arxiv.org/pdf/1507.06527.pdf

37. Bellemare MG, Dabney W and Munos R. A distributional perspective on reinforcement learning. arXiv preprint arXiv:1707.06887 2017.

38. Dabney W, Ostrovski G, Silver D, et al. Implicit quantile networks for distributional reinforcement learning. arXiv preprint arXiv:1806.06923 2018.

39. Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms. In: 31st International Conference on Machine Learning, ICML 2014. Beijing.

40. Heess N, Wayne G, Silver D, et al. Learning continuous control policies by stochastic value gradients. In: Advances in neural information processing systems, 2015, pp. 2944–2952. DOI: 10.48550/arXiv.1510.09142.

41. Schulman J, Heess N, Weber T, et al. Gradient estimation using stochastic computation graphs. In: Advances in neural information processing systems, 2015, pp. 3528–3536.

42. Levine S, Finn C, Darrell T, et al. End-to-end training of deep visuomotor policies. J Mach Learning Res 2016; 17: 1334–1373.

43. Deisenroth M and Rasmussen CE. Pilco: a model-based and data-efficient approach to policy search. In: Proceedings of the 28th international conference on machine learning.
44. Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients. arXiv preprint arXiv:180408617 2018.

45. Lowe R, Wu YI, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in neural information processing systems, December 2017, pp. 6379–6390.

46. Yang Y, Luo R, Li M, et al. Mean field multi-agent reinforcement learning. arXiv preprint arXiv:180205438 2018.

47. Li S, Wu Y, Cui X, et al. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In: Proceedings of the AAAI conference on artificial intelligence, 2019, vol. 33, pp. 4213–4220. DOI: 10.1609/aaai.v33.i01.33014213.

48. Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients. arXiv preprint arXiv:170508926 2017.

49. Li S, Wu Y, Cui X, et al. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In: Proceedings of the AAAI conference on artificial intelligence, 2019, vol. 33, pp. 4213–4220. DOI: 10.1609/aaai.v33.i01.33014213.

50. Hsu CD, Jeong H, Pappas GJ, et al. Scalable reinforcement learning policies for multi-agent control. In: 2021 IEEE/RSJ international conference on intelligent robots and systems (IROS), Prague, 27 September–1 October 2021.

51. Whiteson Research Lab. Python marl framework, 2019, https://github.com/oxwhirl/pymarl

52. Whiteson Research Lab. Python marl framework, 2019, https://github.com/oxwhirl/smac

53. Wang T, Gupta T, Mahajan A, et al. Rode: learning roles to decompose multi-agent tasks. arXiv preprint arXiv:201001523 2020.

54. Silver D, Hubert T, Schrittwieser J, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 2018; 362: 1140–1144.

55. Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge. Nature 2017; 550: 354–359.

56. Zerbel N and Yliniemi L. Multiagent Monte Carlo tree search. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems, Montreal, QC, Canada, 13–17 May 2019, pp. 2309–2311. Richland, SC: International Foundation for Autonomous Agents and MultiAgent Systems.

57. Yee T, Lisy V, Bowling MH, et al. Monte Carlo tree search in continuous action spaces with execution uncertainty. In: IJCAI, 9–15 July 2016, New York, pp. 690–697. Palo Alto, CA: AAAI Press.

58. Lee J, Jeon W, Kim GH, et al. Monte-Carlo tree search in continuous action spaces with value gradients. In: Proceedings of the AAAI conference on artificial intelligence, 2020, pp. 4561–4568. DOI: 10.1609/aaai.v34i04.5885.

59. US Army Training and Doctrine Command. The U.S. Army in multi-domain operations 2028, 2021, https://www.army.mil/article/243754/the_u_s_army_in_multi_domain_operations_2028

60. Headquarters Department of the Army. Army doctrine publication ADP 1-02: terms and military symbols. Scotts Valley, CA: CreateSpace Independent Publishing Platform; 2018.
of situation awareness, automated information extraction for human comprehension, and human-robot collaboration through reinforcement learning simulations.

**Christopher D Hsu** is a research scientist at DEVCOM Army Research Laboratory interested in learning, planning, and control for multi-agent systems. He received his B.S. degree in Mechanical Engineering from Villanova University in 2017 and is currently an Electrical and Systems Engineering Ph.D. student and member of the GRASP Laboratory at the University of Pennsylvania where he also received his M.S.E. in Robotics in 2020.

**Michael R Dorothy** received the B.S. degree in aerospace engineering from Iowa State University in 2009 and the Ph.D. degree in aerospace engineering in 2016. He has been an Aerospace Engineer for DEVCOM Army Research Laboratory since 2015. His research interests include nonlinear dynamics, switched systems, sensitivity-based control, differential game theory, multi-agent collaborative and adversarial planning.

**John G Rogers III** received B.S. and M.S. degrees in Electrical and Computer Engineering from Carnegie Mellon University in 2002, a M.S. in Computer Science from Stanford in 2006, and a Ph.D. in Robotics from Georgia Tech in 2013. He has been a research scientist at the DEVCOM Army Research Laboratory since 2013 and a senior research scientist since 2020. His research interests include multi-agent distributed state estimation, navigation in difficult off-road terrain, and multi-agent collaborative adversarial tactical maneuver.

**John M Fossaceca**, Ph.D., is the Program Manager for the Army Research Laboratory’s Essential Research Program in Artificial Intelligence for Maneuver & Mobility, where he supports research programs in machine learning and artificial intelligence related to autonomous systems maneuver and situational understanding. The AIMM-ERP is operationalizing the science AI-enabled systems for autonomous maneuver that can rapidly learn, adapt, reason, and act in multi-domain operations. Prior to joining ARL in 2016, Dr. Fossaceca worked in industry for several DoD and Telecommunications companies in numerous leadership roles. Dr. Fossaceca also serves as an adjunct professor at GWU teaching courses in machine learning and cyber security.