Contested logistics simulation output analysis with approximate dynamic programming: a proposed methodology

Matthew Powers
Operations Research, The MITRE Corporation, Washington, District of Columbia, USA, and
Brian O'Flynn
Division of Simulation, Experimentation and Gaming, The MITRE Corporation, Washington, District of Columbia, USA

Received 25 July 2022
Revised 27 October 2022
31 October 2022
Accepted 31 October 2022

Abstract

Purpose – Rapid sensitivity analysis and near-optimal decision-making in contested environments are valuable requirements when providing military logistics support. Port of debarkation denial motivates maneuver from strategic operational locations, further complicating logistics support. Simulations enable rapid concept design, experiment and testing that meet these complicated logistic support demands. However, simulation model analyses are time consuming as output data complexity grows with simulation input. This paper proposes a methodology that leverages the benefits of simulation-based insight and the computational speed of approximate dynamic programming (ADP).

Design/methodology/approach – This paper describes a simulated contested logistics environment and demonstrates how output data informs the parameters required for the ADP dialect of reinforcement learning (aka Q-learning). Q-learning output includes a near-optimal policy that prescribes decisions for each state modeled in the simulation. This paper's methods conform to DoD simulation modeling practices complemented with AI-enabled decision-making.

Findings – This study demonstrates simulation output data as a means of state–space reduction to mitigate the curse of dimensionality. Furthermore, massive amounts of simulation output data become unwieldy. This work demonstrates how Q-learning parameters reflect simulation inputs so that simulation model behavior can compare to near-optimal policies.

Originality/value – Fast computation is attractive for sensitivity analysis while divorcing evaluation from scenario-based limitations. The United States military is eager to embrace emerging AI analytic techniques to inform decision-making but is hesitant to abandon simulation modeling. This paper proposes Q-learning as an aid to overcome cognitive limitations in a way that satisfies the desire to wield AI-enabled decision-making combined with modeling and simulation.

Keywords Simulation modeling, Approximate dynamic programming, Contested logistics

Paper type Research paper

Introduction

Current, near peer, adversarial applications of lethal and nonlethal anti-access/area-denial (A2/AD) capabilities expand contested logistic environments and potentially impact mission success. Future joint forces will not enjoy the unhindered global resource access that enables logistic operations support to maritime, land or air supremacy. Future enemy A2/AD tactics will threaten transiting or functioning joint forces through interdiction or by holding intermediate staging bases at risk, thereby denying the use of preferred bases and forcing them to establish longer lines of transit. Strategic diplomatic agreements, or economic and escalatory pressure, will also hinder basing or overflight rights. Logistics forces, supplies or facilities are lucrative and vulnerable targets, increasing the attrition risk of logistics capabilities. As such, if aerial and seaports of debarkation (A/SPOD) are denied, then strategic or operationally distant maneuvering may be required, which further complicates logistic operations (joint concept for logistics).

Simulation modeling of complicated logistic operations plays a vital role in supporting joint forces that are able to project global air, land and sea power. These models include competitive peacetime environments against peer adversaries to achieve dominance in all conflict domains. These virtual environments enable interaction with simulated systems and networks to gain insight into logistic decision-making impacts and to seek better solutions (MacFarland, 2021).

Simulation modeling seeks better solutions without costly prototyping or live exercises. This motivates experiments with simulation input parameters and simulation output analysis. In this spirit, we propose a method that combines simulation modeling with approximate dynamic programming (ADP) as an output analysis method to optimize the decisions under scrutiny in the simulated environment. High-level simulation output analyses are demanding and time consuming due to the considerable amount of complex output data that grows exponentially with the number of units, locations, sensors or weapons. Depending on the model being used, or the scope of the environment being modeled, a single run may generate gigabytes (GBs) of data that demand rigorous analysis to produce useful insights. The analysis must identify trends across multiple simulation runs to understand how decisions affect the logistic operations. Furthermore, these analyses inform timely, high-level, mission-critical decisions, so turnaround between model construction and output analysis is, by necessity, rapid.

Rapid turnaround and quick sensitivity needs to make ADP an attractive method for simulation output analysis. Here, we describe a method where simulation output data informs the parameters of an ADP model known as Q-learning, or reinforcement learning. We describe a simple contested logistics environment, manifested in Simio, and the degree to which the output data helps parameterize the states, actions, contributions and state–state transitions required for ADP. The ADP model output includes an approximately optimal policy that prescribes decisions for each state modeled in the simulation. This method enjoys the benefits of rapid simulation analysis with the desire to wield AI-informed decision-making, all while incorporating the joint force paradigm of modeling complex, adversarial environments with simulation modeling. This is an important and attractive feature of our method. The United States military is eager to embrace emerging analytic techniques such as AI decision-making aids but is rightly hesitant to completely abandon simulation modeling.

The following section provides background on both contested logistic operations simulation modeling and ADP. Next, we discuss literature related to these same topics followed by describing methodology specific to a simple scenario modeled in Simio, and the ADP parameters. We then discuss experimental results and their validation. We conclude this paper by describing contributions to the world of contested logistics modeling and decision-making.

Background

Simulation modeling contested logistics

Simulation modeling of contested logistics, or for any military logistics context, has a rich history of aiding decision-making. Simulation has transformed from a brute-force numerical integration method into an attractive (and preferred) option for decision-makers. This is due to the exponential growth of computing power combined with the ability to harness vast amounts of data and logistic requirements into a structured system. In this, and particularly in military decision-making contexts, simulation is frequently considered a "first resort" for complex real-world issues (Powers *et al.*, 2012).

Consider the model of an air defense system from Rich (1955), where the combination of decision-makers and simple emulation devices comprises the simulation of the dynamic environment. This demonstrates the rich history of simulation as a military decision-making aid. Today, the United States Department of Defense (DoD) regularly apply wide varieties of large simulations to assess how forces should be equipped, organized, trained, employed, deployed, and maintained. Some large scale simulations, such as the Synthetic Theater Operations Research Model (STORM), may be outpacing "Lucas's Law," that is: "The detail within a model grows in proportion to the processing power available, leaving runtimes relatively constant" (Powers *et al.*, 2012). A STORM scenario, like most common, simulation scenarios, is a complex system with big data challenges that may yield hundreds of millions of possible actions. As such, output analysis of these large-scale simulations benefits from complementary optimization methods such as ADP.

Approximate dynamic programming

ADP is a sequel to Richard Bellman's 1950 introduction of dynamic programming (DP). DP is the recursive, iterative process of discovering optimal strategies for dynamic, sequential and uncertain decision-making problems (Rust, 2019). The stochastic version of Bellman's equation sets the stage for the learning version that is applied in this study. The applied ADP dialect of Q-learning engages an AI agent to learn in a dynamic environment. The agent, starting from a state s and performing an action a, transitions to a new state while being rewarded, with the objective of maximizing the reward while mapping system states to action spaces to obtain a high quality decision-making policy. We emphasize the *high-quality* optimality approximation Q-learning solution as an important feature for solving intractable, high-dimension state–space vector problems (Alkaabneh *et al.*, 2021). Q-learning parameters are state–space S, actions A, contribution function C and state–state reachability vice transition probability matrices (TPMs). State–state reachability *in lieu* of TPMs is a nice feature for real-world problems where transition probability knowledge is unknowable or the state–space makes transition probability calculation intractable. Q-matrix $Q(s, a)$ holds the expected reward for each possible action a taken from state s. $Q(s, a)$ achieves band-convergence, allowing Q-learning maximum expected reward and optimal policy R^* approximation by defining the value of each state, $V(s)$, according to (1):

$$V(s) = \max_{a} Q(s, a) \tag{1}$$

At iteration n, $Q(s,a)$ populates with $\widehat{q^n}$ approximated by (2):

$$\widehat{q^n} = \widehat{C}(s^n, a^n) + \gamma \max_{a' \in A} \overline{Q^{n-1}}(s^{n+1}, a') \tag{2}$$

In (2), $Q(s,a)$ convergence requires discount parameter γ, which values a future reward as if it were current reward when set close to 1.0, or prefers short-sighted, myopic optimal policies when set close to 0.0. With (2), the AI agent learns optimal action a from each state s by defining optimal a at iteration n with (3):

$$a^n = \operatorname{argmax}_{a \in A} \overline{Q^{n-1}}(s^n, a) \tag{3}$$

Q-learning uses the estimates $\overline{Q^{n-1}}(s^n, a)$ from iteration *n-1* at iteration *n*. With $\widehat{q^n}$, Q-factors in *Q(s,a)* update via the learning version of Bellman's, as seen in equation (4):

$$\overline{Q^n}(s^n, a^n) = (1 - \alpha_{n-1})\overline{Q^{n-1}}(s^n, a^n) + \alpha_{n-1}\widehat{q^n} \tag{4}$$

This model sets equation (4) learning rate parameter α to 0.1, avoiding Q-factor *apparent* convergence (since $\alpha > 0.0$) without learning too quickly (since $\alpha < 0.99$). Equation (4) derives total reward $\Sigma V(s)$, and optimal policy vector R^*, which describes approximately optimal decision-making strategies in our contested logistics environment.

Literature review

Simulation and contested logistics

The rich history of simulation modeling to aid logistic decisions is recognized in MacFarland (2021), which focuses on simulation experiments with goals to assess joint interoperability, adversarial deterrents, or validating plans. In this spirit (Ghanmi *et al.*, 2008), applies discrete event simulation to assess Multinational Intra-Theater Distribution (MN ITD) coalition operation logistics distribution systems. However, logistic decision-making simulation models are not monopolized by military context. For example Vidalakis *et al.* (2011), presents simulation modeling of construction material distribution logistics through intermediate nodes similar to the various locations in our contested logistics scenario (Réveillac, 2017); focuses on logistic system flow simulation to generate data that are collected and used to build optimization models across software ranging from flow simulators to simple spreadsheets, similar to our study's use of Simio data to build an ADP model (Dehghanimohammadabadi and Keyser, 2015); combines simulation and optimization by calling the optimizer during simulation execution processes applicable to supply-chain and machine maintenance problems. Similarly, our method recommends validating ADP approximate optimal policies via the contested logistics simulation model scenario.

Approximate dynamic programming

The ADP dialect of Q-learning works well for problems with a relatively small state space, as recognized by Jiang *et al.* (2014), which discusses the shortcomings of lookup tables in large scale applications. However, modern computing speed and the relatively small scale of our contested logistics scenario make Q-learning a tractable technique. Furthermore, the Q-learning value function *Q(s, a)* for state *s* and action *a*, while not as accurate as value iteration, is sufficient to model decision-maker choices.

This study's applied Q-learning R package, *ReinforcementLearning*, refers to Sutton and Barto (2018) for its technical and theoretical details. The authors emphasize that Q-learning band-convergence satisfies most real-world decision-making requirements akin to those in this contested logistics study. The importance of real-world applicability is highlighted in *ReinforcementLearning* literature from Pröllochs and Feuerriegel (2018). Band-convergence is a satisfactory optimal approximation according to an assumption stated by Silver *et al.* (2017), wherein Q-learning requires a solid base of first principles that need not include state transition probabilities. Our contested logistics "first principles" refer to the action–reward and state–state reachability parameters evaluated in a simulated environment. Our military simulated environment resembles those seen in Summers and Robbins (2020) and Davis *et al.* (2017), both involving ADP-derived approximately optimal sequential missile engagement strategies. These strategies are solutions in high-dimension state–spaces that make alternative solution methodologies intractable. For real-world, high-dimension problems, traditional optimization methods are not possible due to long computation time and memory requirements. This is known as the curse of dimensionality (Powell, 2011). Our model captures sequential decision-making in a tractable contested logistics context, demonstrating a similar need for optimal policies in a complex, dynamic environment.

Uncertain dynamic environments motivate ADP MEDEVAC dispatch policies as seen in Nasrollahzadeh *et al.* (2018), Robbins *et al.* (2020) and Rettke *et al.* (2016). Nasrollahzadeh *et al.* (2018) recognizes ADP ability to compute high quality solutions in unbounded state–spaces, while Robbins *et al.* (2020) goes on to recognize the benefit of ADP as a framework to compute high quality policy approximations with less computational expense than a high-fidelity scenario-based simulation. Our contested logistics model reconciles ADP vs. simulation modeling via a complementary approach that resembles (Astaraky and Patrick, 2015), wherein the authors apply the ADP concept of post-decision state variables to a surgical scheduling model that has been informed by simulation. The authors recognize that problems may remain intractable despite ADP state–space reductions and recommend a function approximation architecture where a class of functions parameterizes state values *in lieu* of Q-learning lookup tables. In a similar surgery scheduling context (Silva and de Souza, 2020), recommend approximate policy solutions in the face of dynamic decision-making environments that anticipate new information. Value function approximation is tenable via approaches akin to regression analysis of simulation output data representing state values, should the scenario grow beyond the complexity modeled in this study.

Methodology

Scenario description

Our notional scenario, although fictional and not aligned with logistics doctrine or tactics, techniques and procedures (TTPs), represents a useful contested logistics challenge. A supported unit (TF1) and a supporting aerial port are located within an area of responsibility (AOR) that is vulnerable to red adversary Dong–Feng 21 (DF-21) medium-range ballistic missile launchers. Aerial port operations are defined by velocity and volume of supplies needed for TF1. TF1 moves deeper into adversary territory, becoming vulnerable to additional red units, artillery and air defense artillery (ADA). The aerial port must depart its current location at edge port 1 (EP1) and routinely move to other EPs inside the AOR to avoid DF-21 targeting while continuing logistics support (defined by velocity and volume of supplies) to TF1. Figure 1 is a Simio screenshot that summarizes the scenario.

The scenario's operational decisions, which must be made frequently, are to which potential aerial port location should the EP move, what type of transportation should the EP use to get there, and how should the EP perform resupply logistic support operations to TF1 at its new location. Consider the combinatorial complexity of the following decision parameters:

(1) A total of 19 potential aerial port locations.

(2) A total of 7 possible movement combination options of aircraft, ship or truck.

(3) A total of 7 possible resupply combination options of drone, ship or truck.

These decision parameters, alone, naively yield 931 options, each of which possesses pros and cons associated with movement costs and distances, resupply costs and distances and threat vulnerability. For example, the potential aerial port location (blue square) located immediately south of the supported unit's new location (purple square) in Figure 1 seems ideal. It is outside the red threat range, and it is close to the supported unit; however, the EP cannot always remain at that location because of vulnerability to DF-21 targeting, and movement options from that location are costly in terms of vulnerability or distances to other movement options. But should the EP opt to move to a location that is further away from the supported unit, resupply operations are more costly. These are the types of considerations

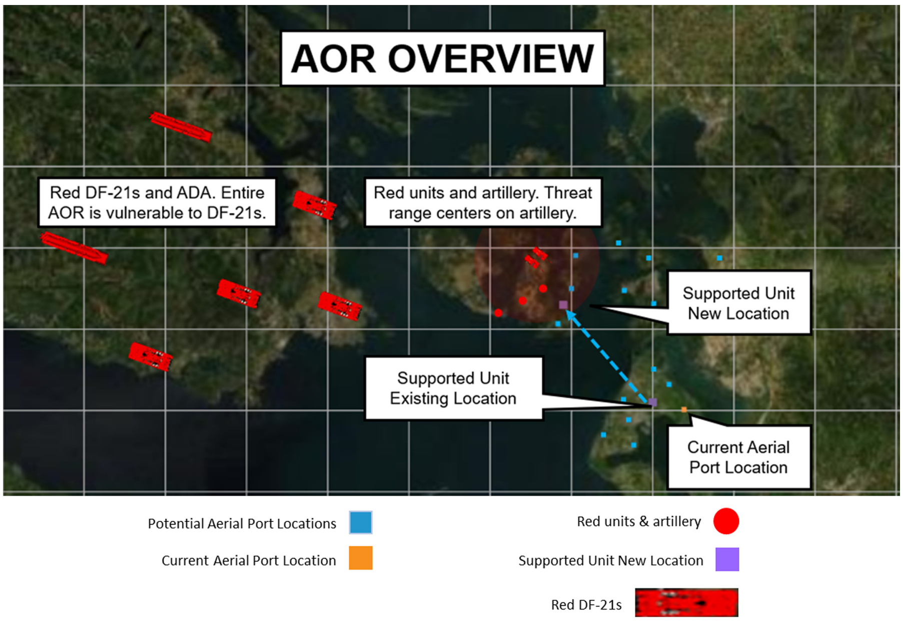

AOR OVERVIEW
Red DF-21s and ADA. Entire AOR is vulnerable to DF-21s.
Red units and artillery. Threat range centers on artillery.
Supported Unit New Location
Supported Unit Existing Location
Current Aerial Port Location
Potential Aerial Port Locations
Current Aerial Port Location
Red units & artillery
Supported Unit New Location
Red DF-21s

Figure 1. Simio screenshot scenario's summary

that must be given to each decision. In this, even simple scenarios present large decision space problems that despite the aid of simulation modeling can overwhelm the decision-maker with descriptive simulation output data that require rigorous, complementary simulation output analysis. The following sections describe how Simio modeling reduces the decision space so that the ADP output analysis does not suffer the curse of dimensionality.

Simio modeling

Simio software provides a workspace for modeling facility resources and entities, as well as for running discrete event simulation (DES). It provides a standard library of fixed objects and the tools for developing accurate 3D models that represent process workflows and resources. Simio also supports animating workflows and the integration of custom models representing shop floor or aerial port resources into animations.

Sophisticated aerial port operations, material handling workflows, and process logic may be modeled and animated using DES software such as Simio. The software can model the operation of a system as a discrete sequence of events in time. Each event occurs at a particular instant and marks a change of state in the system. For example, an aerial port activity that includes packing crates, loading pallets, moving pallets via k-loaders, forklifts and aircraft, can be defined and modeled in Simio as a sequence of discrete events or activities. In addition, the process logic associated with those activities may be modeled. For the ADP-Simio application, Simio will model the velocity and volume of supplies needed on a daily basis to sustain TF1 at its new location, defined as *TF1′*.

Approximate dynamic programming modeling

The TPM-free ADP dialect of Q-learning learns near-optimal behavior through random interactions within a dynamic environment and reward-based performance evaluation. Q-learning is unsupervised as there are no specific improvement behavior instructions.

Q-learning deals with unstructured state–spaces possessing unknown transition probabilities with a learning version of Bellman's equation that converge to a near-optimal band objective vice a point-value. Q-learning is attractive for this study because of the relatively small state–space, sufficient computing power and the lack of TPM requirement.

We parameterize our Q-learning model with states S, actions A and rewards $C(s^n, a^n)$. As seen in (2), $Q(s,a)$ convergence requires discount parameter γ, which is the value of future rewards. Setting γ close to 1.0 treats future rewards as current rewards, while γ close to 0.0 prefers immediate reward and a myopic near-optimal policy. There is opportunity to experiment with model sensitivity to $0.0 \leq \gamma \leq 1.0$, but for this study we set $\gamma = 0.5$. We use the R package *ReinforcementLearning* for Q-learning application, which attributes theoretical and technical details to Sutton and Barto (2018). The authors recognize that real-world problems benefit from a constant learning rate α in (4) for rapid band-convergence. As such, we set α to 0.1, avoiding Q-factor apparent convergence (since $\alpha > 0.0$) without learning too quickly (since $\alpha < 0.99$). *ReinforcementLearning* requires a data frame input in a state/action/reward/next-state format to represent the stochastic environment with inter-state reachability. We refer to Figure 2 Simio screenshot for states, actions, contributions and transitions, with expository descriptions in the sections that follow.

States. We define a state at iteration n, $s^n \in S$, as a combination of location, how the aerial port had arrived at the location, and how the aerial port provides resupply. Simply put, the aerial port's state describes where it is, how it got there and what it is doing.

Simio model operations analysis reduces possible EP location options to $EP' = \{$EP1, EP13, EP14, EP15 and EP17$\}$. EP1—the orange square in Figure 2—is the starting location in the scenario from which the aerial port does not return after departing to a different location. The remaining feasible EPs are Figure 2 blue squares located within the yellow-dashed air defense cover and within approximately eight miles of the supported unit's new location $TF1'$.

The current state is also defined by how the aerial port arrived at the current location. This state variable assumes that if the aerial port used a resource to arrive at its current location, then it is less costly to use that resource to move. We define this state variable *REL* via seven possible movement combination options of aircraft, ship or truck.

Finally, the current state is defined by how the aerial port is providing resupply. This state variable assumes that if the aerial port used a resource to provide resupply, then it is less costly to use that resource in some future state. We define this state variable *RES* via seven possible resupply combination options of drone, ship or truck. In this, $S = \{EP', REL \text{ and } RES\}$.

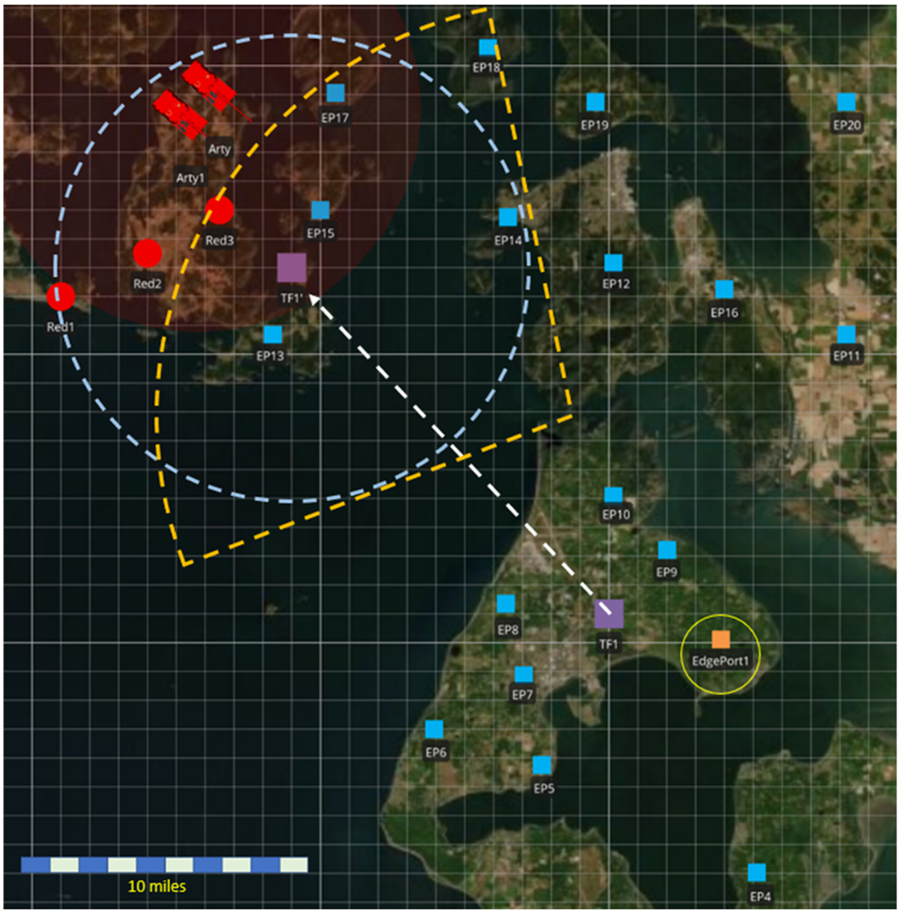
We codify each current and future state as a binary vector in the set order of $\{EP', REL \text{ and } RES\}$. For illustration, the state of being at EP13, having moved by aircraft, and resupplying via drone and truck codifies as {01000 100 101}. Binary representation eases coding and reduces computational complexity between state, action, reward and transition interactions.

Actions. We define an action a at iteration n, $a^n \in A$, as a combination of next location, how the aerial port will move and how the aerial port will provide resupply. Possible next location options are $EP'' = \{$EP13, EP14, EP15 and EP17$\}$.

We define movement actions REL' as seven possible movement combination options of aircraft, ship or truck.

Similarly, we define resupply actions RES' as seven possible resupply combinations options of drone, ship, or truck.

In this, $A = \{EP'', REL' \text{ and } RES'\}$. All chosen actions are codified as binary vectors, similar to state variable representations. The costs and benefits of each action contribute to the reward for each decision made from each state.



Potential Aerial Port Locations	🟦	Air Defense Cover	— — —
Current Aerial Port Location	🟧	Potential Aerial Port location restriction (within 8 miles of TF1′)	— — —
Red units & artillery	●	Red DF-21s	🟥
Supported Unit New Location	🟪		

Figure 2.
Simio screenshot for states, actions, rewards and transitions

Rewards or contributions. Reward value construction begins with calculating weights ω associated with location decision distances to threats RT = {red1, red2, red3 and artillery}, $TF1'$, REL' and RES'.

Weights $\omega_D(i)$, $i \in \{1 \ldots 5\}$ associate with location distances to $\{RT, TF1'\}$.

Weights $\omega_R(j)$, $j \in \{1 \ldots 6\}$ associate with REL' and RES' costs.

Next, we construct vulnerability weights υ based on REL' and RES' vulnerability to RT. We normalize these weights, $\omega_D(i)'$, $\omega_R(j)'$ *and* $\upsilon(j)'$ with the scheme $\omega_D(i)' = \omega_D(i)/\Sigma_{\omega\upsilon}$ for all weights, where $\Sigma_{\omega\upsilon} = \text{sum}(\omega_D(i), \omega_R(j), \upsilon(j))$.

Distance matrix D_{km} stores distances between k and m where k and $m \in K = \{EP'', TF1'$ *and* $RT\}$. We initialize red threat reward coefficient vector $RT_{rew} = \{1, 1, 1, 1\}$. The reward for moving away from $RT(m)$, $m \in \{1 \ldots 4\}$ is $RT_{rew}(m) = 0$, which is evaluated with D_{km}. Similarly, we initialize supported unit reward coefficient $TF_{rew} = 0$, and $TF_{rew} = 1$ is the reward for moving toward $TF1'$.

We define action $\hat{a}^n$ as the binary $\{REL', RES'\}$ vector subset representation of the action taken at any iteration n, and D_a as the distance between EP_{n-1}'' and $EP_{n'}''$. With these, we define the movement and resupply cost, C_{R2}, with (5).

$$C_{R2} = D_a * \sum_j \omega_R(j)' * \hat{a}^n / \dim(\hat{a}^n) + \sum_j \omega_R(j)' * (\hat{a}^n - \{REL, RES\}) \tag{5}$$

Equation (5) considers C_{R2} as a function of distance and of switching movement methods between states. Similarly, (6) defines the reward for action vulnerability, C_v.

$$C_v = D_a * \sum_j v(j)' * \hat{a}^n / \dim(\hat{a}^n) \tag{6}$$

Note that (6) represents a vulnerability window for time in transit, not for distance to threats, which is accounted for in (7), the total reward $\hat{C}$.

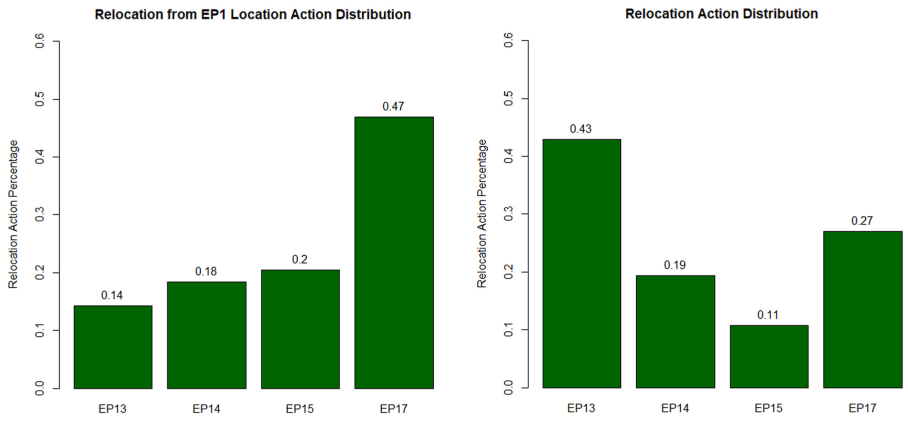
$$\hat{C} = \sum_{i,m}^{4} \omega_D(i) * RT(m) + \omega_D(5) * TF_{rew} - C_{R2} + C_v \tag{7}$$

Transitions. Q-learning replaces TPMs with state–state reachability in a state space explored by the agent to learn the approximately optimal policy. To this end, *ReinforcementLearning* requires a data frame of current state, action, reward and next state. We initialize this data frame via naïve random sampling of all possible combinations, and then we refine the data frame by removing state – next-state entries that violate scenario "rules." For example, note that in Figure 2, EP13 is south of $TF1'$. Should the current state location be EP13, and the movement action be to move to EP15 (located north of $TF1'$), then we assume that no amount of uncertainty would result in the aerial port ending up in EP14 at the next state. Therefore, any data frame row that contains current state EP13, action move to EP15 and next state EP14, removes from the state space. Similar rules may apply to other movement or resupply actions as suitable for the scenario. This method has the benefit of state space reduction and is an adaptable way to tailor a scenario for "what if" analyses.

Experimental results

Simio-ADP contested logistics model results are too diverse to display completely, but suffice it to say that ADP policies present decisions that are not obvious or intuitive. This is because ADP considers sequential decisions under uncertainty in timelines that extend far beyond human cognitive capability. Here, in Figure 3, we present results with assigned ADP parameter values that align with those in the Simio scenario to demonstrate how ADP produces prescriptive outcomes to aid the decision-maker. In this, our model serves as a prototype for a dynamic method for making approximately optimal decisions in a rapidly changing operational environment. Decision-makers need not be alarmed by the approximation of optimality vice optimal decisions in large-scale problems where true optimality is intractable. Q-learning iterates until apparent convergence, thus guaranteeing high quality policies in an unsolvable optimality problem. For example, Figure 3 displays nearly optimal location actions when departing EP1 throughout the scenario.

Consider the less-than-obvious policy that recommends departing EP1 for EP17 most of the time, as seen in Figure 3 left plot. EP17 is the furthest option from EP1, is among the

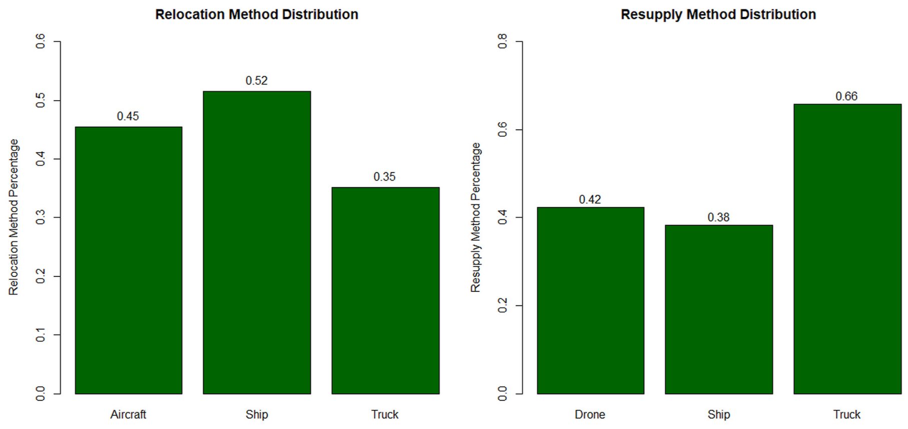


Note(s): These are the actions regarding which location to move to upon making a decision. For illustration, ADP results say that the aerial port should depart EP1 for EP17 47% of the time. However, after departing EP1, the aerial port will only move to EP17 27% of the time

Figure 3. Movement action distribution from EP1 (left plot) and throughout the scenario (right plot)

furthest from *TF1′* and is located within the red threat ring (see Figure 2). It is not obvious that EP17 would be the most frequently recommended initial location from EP1 in a near-optimal policy. This illustrates that ADP considers factors beyond the myopic, greedy choice that may have most often recommended EP13 as the best initial option. However, note that EP13 is the most often recommended location option throughout the course of the scenario at 43%, according to Figure 3 right plot. Once again, ADP considers a long time horizon and state variables such as movement and resupply methods, which are described in Figure 4.

The plots in Figures 3 and 4 are useful summary statistics of near-optimal actions, but a dynamic contested logistics environment benefits from prescriptive analysis outputs such as



Note(s): Note that, for both plots, the sum of distribution percentages are greater than 1.0 because some of the method options include combinations of more than one method. We see that the most common movement methods include ships, and the most common resupply methods include trucks

Figure 4. Movement and resupply method distributions throughout the scenario

the ADP policy. To this end, we develop a model query function that yields output examples like those seen in Figure 5, which details the first six rows of a data frame of all actions that include relocating to EP13 with methods that include ships and resupplying with methods that include trucks.

The first row in Figure 5 tells us that if we are in a state of being located at EP15 (a "1" under the "EP15" column), having had moved by aircraft ("1" under "relAC"), and having been resupplying via drone and ship (a "1" under "resDrone" and "resShip"), then we should move to EP13 by way of ship, after which we should resupply via drone, ship, and truck (a "1" under columns "A_EP13", "A_relShip", "A_resDrone", "A_resShip" and "A_resTruck"). Similarly, the sixth row tells us that if we are located in EP14, having had moved by aircraft and ship, and having been resupplying with ship and truck, then we should move to EP13 by way of ship, where we should resupply via ship and truck.

The results in Figure 5 are a small sample of decisions that guarantee near-optimal value throughout the scenario timeline. Even with this simple scenario, the decisions are not always intuitive, and the reasoning behind them spring from the reinforced learning of the ADP agent, who experiences consequences for the decisions made on a long time horizon. Of course, these results reflect the parameter values in this model instance, but an attractive ADP feature is the speed with which it enables experimenting with parameter values to compare results.

Results validation

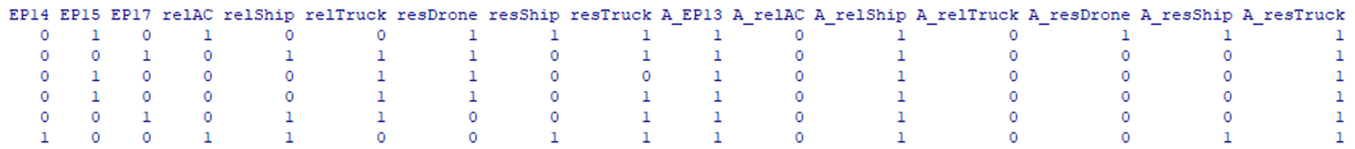
ADP used in combination with Simio and other federated models offers new insights into AI-enabled, logistics-centric decision-making. MITRE plans to validate the Simio-ADP results, in coordination with various sponsors, in a campaign-level model that will track velocity and volume of supplies delivered in an adaptive scenario.

Contributions and conclusions

This paper contributes a methodology that conforms to DoD simulation modeling practices complemented with AI-enabled decision-making to evolve anecdotal, experience-based, or antiquated methods. The military recognizes AI as a necessary resource to keep pace with near-peer adversaries, yet stubbornly sticks to decision-making processes that do not adapt to dynamic environments over lengthy horizons. Lettau and Uhlig (1999) ponder this phenomenon:

> It is intriguing to speculate about possible resolutions such as instincts, learning from your peers, education, meta-rules for changing rules, or the neuronic limits of the brain. We simply take these given rules, as well as the fact that the agent stubbornly sticks to choosing between them throughout his infinite life as primitives of the environment.

Regardless of the reasons behind stubborn reliance on reasoning methods that do not keep pace with the dynamics of a complex environment, all of them are vulnerable to the misconception of "boundless rationality," the suggestion that humans possess unbounded levels of rational, computational ability (Rust, 2019). In reality, humans are unlikely to grasp optimal or approximately optimal strategies in uncertain, long time horizons. Here, we demonstrate ADP as an aid to overcome these limitations in a way that satisfies the desire to wield AI-enabled decision-making that complements established modeling and simulation methodology. This study combines simulation output data as a means of state–space reduction to avoid the curse of dimensionality. Further research would benefit from using simulation output data to estimate Q-value approximations, thereby setting the stage for similar near-optimal policy requirement problems on a larger scale.



```
EP14 EP15 EP17 relAC relShip relTruck resDrone resShip resTruck A_EP13 A_relAC A_relShip A_relTruck A_resDrone A_resShip A_resTruck
   0    1    0     1       0        0        1       1        1      1       0         1          0          1         1          1
   0    0    1     0       1        1        1       0        1      1       0         1          0          0         0          1
   0    1    0     0       0        1        1       0        0      1       0         1          0          0         0          1
   0    1    0     0       0        1        1       0        1      1       0         1          0          0         0          1
   0    0    1     0       1        1        0       0        1      1       0         1          0          0         0          1
   1    0    0     1       1        0        0       1        1      1       0         1          0          0         1          1
```

Figure 5.
Model query output for movement to EP13 via ship, resupply with truck actions (first six rows)

References

Alkaabneh, F., Diabat, A. and Gao, H.O. (2021), "A unified framework for efficient, effective, and fair resource allocation by food banks using an Approximate Dynamic Programming approach", *Omega*, Vol. 100, 102300.

Astaraky, D. and Patrick, J. (2015), "A simulation based approximate dynamic programming approach to multi-class, multi-resource surgical scheduling", *European Journal of Operational Research*, Vol. 245 No. 1, pp. 309-319.

Davis, M.T., Robbins, M.J. and Lunday, B.J. (2017), "Approximate dynamic programming for missile defense interceptor fire control", *European Journal of Operational Research*, Vol. 259 No. 3, pp. 873-886.

Dehghanimohammadabadi, M. and Keyser, T.K. (2015), "Smart simulation: integration of Simio and MATLAB: 2", in *Proceedings of the 2015 Winter Simulation Conference*.

Ghanmi, A., Campbell, G.B. and Gibbons, T.A. (2008), "Modeling and simulation of multinational intra-theatre logistics distribution", *2008 Winter Simulation Conference*, pp. 1157-1163.

Jiang, D.R., Pham, T.V., Powell, W.B., Salas, D.F. and Scott, W.R. (2014), "A comparison of approximate dynamic programming techniques on benchmark energy storage problems: does anything work?", *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 1-8.

Lettau, M. and Uhlig, H. (1999), "Rules of thumb versus dynamic programming", *The American Economic Review*, Vol. 89 No. 1, pp. 148-174.

MacFarland, S. (2021), "Joint force experimentation for great-power competition", Heritage Foundation, available at: https://www.heritage.org/military-strength-topical-essays/2021-essays/joint-force-experimentation-great-power-competition

Nasrollahzadeh, A.A., Khademi, A. and Mayorga, M.E. (2018), "Real-time ambulance dispatching and relocation", *Manufacturing and Service Operations Management*, Vol. 20 No. 3, pp. 467-480.

Powell, W.B. (2011), *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, John Wiley & Sons, Incorporated, Hoboken.

Powers, M.J., Sanchez, S.M. and Lucas, T.W. (2012), "The exponential expansion of simulation in research", *Proceedings Title: Proceedings of the 2012 Winter Simulation Conference (WSC)*, IEEE, Berlin, pp. 1-12.

Pröllochs, N. and Feuerriegel, S. (2018), "Reinforcement learning in R", *arXiv:1810.00240 [cs, stat]*.

Rettke, A.J., Robbins, M.J. and Lunday, B.J. (2016), "Approximate dynamic programming for the dispatch of military medical evacuation assets", *European Journal of Operational Research*, Vol. 254 No. 3, pp. 824-839.

Réveillac, J.M. (2017), *Modeling and Simulation of Logistics Flows 2: Dashboards, Traffic Planning and Management*, John Wiley & Sons, Incorporated, Newark.

Rich, R.P. (1955), "Simulation as an aid in model building", *Journal of the Operations Research Society of America*, Vol. 3 No. 1, pp. 15-19.

Robbins, M.J., Jenkins, P.R., Bastian, N.D. and Lunday, B.J. (2020), "Approximate dynamic programming for the aeromedical evacuation dispatching problem: value function approximation utilizing multiple level aggregation", *Omega*, Vol. 91, 102020.

Rust, J. (2019), "Has dynamic programming improved decision making?", *Annual Review of Economics*, Vol. 11 No. 1, pp. 833-858.

Silva, T.A. and de Souza, M.C. (2020), "Surgical scheduling under uncertainty by approximate dynamic programming", *Omega*, Vol. 95, 102066.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. and Lillicrap, T. (2017), "Mastering chess and shogi by self-play with a general reinforcement learning algorithm", *arXiv preprint arXiv:1712.01815*.

Summers, D. and Robbins, M.J. (2020), "An approximate dynamic programming approach for comparing firing policies in a networked air defense environment", *Computers and Operations Research*, Vol. 117, p. 15.

Sutton, R.S. and Barto, A. (2018), *Reinforcement Learning: An Introduction*, 2nd ed., The MIT Press, Cambridge, MA, London.

United States (2015), *Joint Concept for Logistics*, Joint Chiefs of Staff, Washington, D.C, available at: https://www.jcs.mil/

Vidalakis, C., Tookey, J.E. and Sommerville, J. (2011), "Logistics simulation modelling across construction supply chains", *Construction Innovation*, Vol. 11 No. 2, pp. 212-228.

Corresponding author
Matthew Powers can be contacted at: mpowers@mitre.org