26th International Congress on Modelling and Simulation, Adelaide, SA, Australia, 30 Nov–4 Dec 2025
mssanz.org.au/modsim2025

SWAR: Evaluating the Effect of Distributed Communication Approaches on Application Effectiveness

Dumitru-Alin Balasoiu [a], **Ishan Honhaga** [a], **Gregory Judd** [b], **Dustin Wyly Craggs** [a], **Benjamin Campbell** [c], **Claudia Szabo** [a]

[a]*School of Computer Science, The University of Adelaide, South Australia*
[b]*Aurizn*
[c]*Defence Science and Technology Group, Defence, Adelaide, South Australia*
Email: dumitru-alin.balasoiu@adelaide.edu.au

Abstract: The success of critical operations, such as military or disaster and recovery operations, depends on streamlined and effective coordination and collaboration efforts. In contested and dynamic environments, where communication networks are unstable and resources may become unavailable and where environmental conditions change frequently, quantifying the effect of lack of access to high quality, timely information helps decision makers deploy supplementary resources more efficiently.

In this paper, we propose the **S**MARTNet **War**game for **AI** **R**esearch (SWAR) simulation environment described in Figure 1, designed to explore the effect of timely and relevant messages on mission success. Our agent-based model captures mission structures and communication patterns and links these to their effect on the success of specific missions. We demonstrate the utility of such an environment by integrating it with a reinforcement learning (RL) system to optimise the communication strategy of the agents. While a specific static strategy yielded a 60.61% loss rate for friendly forces (blue), outperforming other dynamic strategies that resulted in approximately 76% losses, this outcome underscores the importance of prioritising certain message types. We also observe that Reinforcement Learning based dissemination policies consistently outperformed rule-based static and dynamic policies, particularly in maximising the survivability of friendly agents and the elimination of enemies. The RL policies achieved an average reward approximately 2.3x higher than the best static baseline.

Keywords: *simulation, emergent behaviour, resilient complex adaptive systems, RCAS*

1 INTRODUCTION

During military tactical operations, soldiers and autonomous agents must rely on wireless infrastructure free communications, such as mobile radio-based ad-hoc networks (MANETs)(Kumar and Mishra (2012)). These systems provide an austere and variable communications environment especially as agents move through complex physical terrain. Due to the nature of battle, soldiers might not have a complete view of the position or status of team mates, and thus critical information dissemination decisions that will affect system-wide properties (i.e mission success) must be made with incomplete and unreliable information. In these scenarios it is crucial to understand which information is critical to mission success, the accuracy and timeliness requirements of this information, and to identify circumstances where partial information is more harmful than no information at all. In our previously published work (Craggs et al. (2022); Szabo et al. (2022); Gopalan et al. (2022)) we focused on information dissemination strategies that optimised metrics that measured information accuracy. The limitation of these approaches was that they provided no means of assessing the impact of information accuracy i.e. the information value. This resulted in an inability to trade off the importance of different information types in constrained environments where not all messages could be sent.

In this paper, we propose the **S**MARTNet **W**argame for **AI** **R**esearch (SWAR) simulation and visualisation environment described in Figure 1. This environment enables the exploration of the impact of information dissemination strategies on application level measures of effectiveness in order to evaluate behaviours in a complex adaptive system. We explore the effect that messages and environment disruptions have on system wide desired outcomes. Together with subject matter experts, we define scenarios with high face validity and consider several disruptive elements, such as bandwidth and suboptimal prioritisation strategies. We demonstrate the value of SWAR by evaluating several rule based message prioritisation strategies and then use it to train a number of Reinforcement Learning (RL) policies.

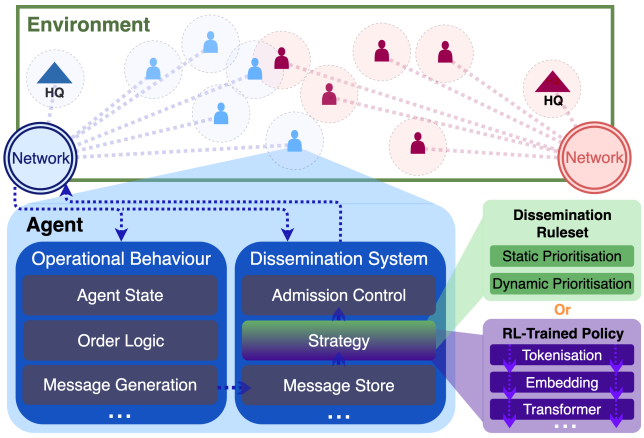

Figure 1. SWAR Overview: Repast Simphony environment with 2 opposing teams: Blue and Red, each team with a fixed commanding agent, HQ, and several Manoeuvre agents that communicate via messages and follow orders to achieve mission objective. Agents communicate using either a static, dynamic or RL-trained dissemination strategy.

In complex multi-agent systems, such as the ones we simulate, emergent behaviours are common and challenging to predict, particularly in adversarial environments with incomplete information. While much existing research on these focuses on the design of systems with specific properties (Jacyno et al. (2009); Salazar et al. (2011); Bernon et al. (2003)) rather than comprehensive evaluation, our work introduces a novel approach of using our proposed simulation testbed, SWAR. Unlike previous studies that rely on easily quantifiable but limited measures of performance (Pumpuni-Lenss et al. (2017); Boyapati and Szabo (2022); Ahmad et al. (2025)), we prioritise measures of effectiveness (Sproles (2002); Eoyang and Berkas (1998)), which provide a holistic, application based view of system behaviour. SWAR is designed to dynamically prioritise information flows based on Quality of Information (QoI) attributes of Suri et al. (2015), a task that becomes particularly complex due to changing mission contexts and unforeseen network disruptions. Our simulation allows us to analyse how low-level agent communication decisions and system-wide protocols impact high-level mission outcomes, such as the survival of friendly forces and the destruction of enemy forces. This analysis, which includes comparisons of static and dynamic policies, represents a crucial step towards developing more collaborative and context-aware agents capable of achieving timely message delivery in complex environments.

To investigate the impact of communication strategies on battlefield outcomes, we implemented a series of experiments comparing static and dynamic policies. Our analysis, detailed in Section 3, provides insights into message prioritisation as the two teams execute their mission objectives and engage in combat. Furthermore, we introduce RL policies utilising a Transformer architecture (Vaswani et al. (2017)) to evaluate the benefits of an adaptive dissemination system. We selected the model-free Q-learning algorithms Deep Q-Networks (DQN)(Mnih et al. (2013)) and QMIX (Rashid et al. (2020)) due to their suitability for complex, high-dimensional state spaces where modelling state transition probabilities is computationally difficult. The

Transformer enables our agents to process the large observation spaces as variable-length sequences of messages and agent representations. This approach is unique in that our RL policy governs message prioritisation, as opposed to the more conventional application of RL to directly control agent behaviour as discussed in Section 4.

2 SWAR Architecture

SWAR models the potential effect of different delays in, or changes in the order of, the receipt of command and control and situation awareness messages on mission outcomes. The simulation analysis allows decision makers to determine the order and rate that different types of messages should be sent in a network contention situation when not all messages can be sent at once.

SWAR is implemented in Repast Simphony 2.11 and adheres to an agent-based simulation paradigm. An overview of the SWAR architechture can be found in Figure 1. Two main agent types are represented in the model, red (enemy) and blue (friendly) agents, and within these two types, agents can be of type Manoeuvre, capable of movement and are responsible for carrying out operational instructions and Headquarters (HQ), is static and responsible for directing Manoeuvre agents. Based on Position Location Information (PLI) messages and local sensing the HQ agent sends orders to all Manoeuvre agents, unless the agent's hit-points are less than a specified threshold in which case it will be ordered to retreat to the HQ location. A class diagram depicting these relationships is shown in Figure 1.

The agents start the simulation at random locations within a specified region in a REPAST space. At the start of the scenario, Manoeuvre agents receive an order to advance towards the enemy HQ. Once the simulation has started, the HQ agent sends orders directing Manoeuvre agents to move closer to enemy agents or retreat. While, Manoeuvre agents execute these orders they relay new operational information to the other agents.

A SWAR simulation scenario allows for the configuration of a large number of parameters including the behaviours of the agents, the type of scenario the agents operate in, and network conditions. Table 1 outlines a few key parameters. The scenario parameters are configured using a JSON file and the scenario can be run either through the Repast Simphony GUI or in headless mode by extending the AbstractRunner class.

Table 1. Description of SWAR parameters.

Parameter	Description
Scenario	Mission and Blue/Red Team configurations
Message/step/agent	Maximum number of messages per agent per step
Sending frequency	Message transmission interval
Communication	Enable Communication amongst the team
Command Usage	Are orders created and sent by the HQ agent
Movement	Flag for enabling Agent Mobility
Can retreat	Can agents be ordered to retreat when in distress
Hit-points	Health of each agent
Distress threshold	Hit-point Threshold for Retreat
Attack distance	Minimum Engagement Distance
Message Categories	Accessible message categories like Agent Deleted, Enemy Location etc.

2.1 Message Types and Prioritisation Rules

Other than the Order message sent by HQ agents, all agents can send the messages in Table 2.

Three message prioritization policies have been defined, namely, *First in First Out (FIFO)*, *static* and *dynamic*. The *FIFO* policy is applied by default when no policies are selected or when no learning modules are activated. For *static* policy the relative priorities of different message categories are predetermined and do not change throughout the run. Whereas, a *dynamic* policy changes the relative priorities of different message categories in response to changes in operational circumstances or available network connectivity or bandwidth.

Table 2. Overview of message types in SWAR

Message Type	Description
Friendly location	Current position of an agent
Enemy location	Position of a detected enemy
Order update	Order that may include location of an enemy agent
Status update	Agent status, including remaining hit-points
Agent deleted	Notification of agent destruction

2.2 Simulation Run

Figure 2 shows the sequence of events that occur at each time step, with a description of the steps below.

1. Update Dissemination Policies: The agent's dissemination prioritisation policy is updated. This update only affects scenarios using dynamic or RL (Reinforcement Learning) prioritisation policies, which can change message priorities based on specific sce-

nario events. In contrast, it has no effect on FIFO or static policies.

2. Send Messages: At each step, an agent prioritises and sends messages until it reaches a configured threshold, defined by the Message/step/agent (see Table 1), or has no messages to send. In the scenario discussed in Section 3, Message/step/agent is one message every two steps.

3. Move: An agent executes received order by moving in a straight line towards the ordered location. Upon arrival to the specified location, the agent stops. However, if the agent is engaged in combat (i.e., within an enemy agent's attack range), it will not execute any new orders except for a retreat order.

4. Resolve Combat: The agent resolves combat by identifying enemies within its attack range. If multiple enemies are present, the agent selects a target based on a configured option, either the closest enemy or a randomly chosen one. Upon engagement, both the attacking and targeted agents lose one hit-point. At the end of this step, any agent whose hit-points have reached zero is removed from the simulation.

5. Terminate: An episode concludes when a pre-specified number of steps has been completed or when one side loses all of its agents.

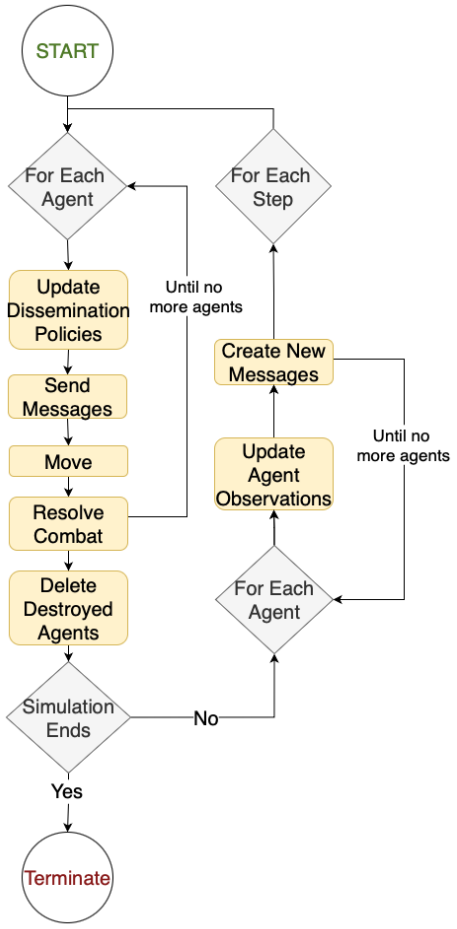

Figure 2. SWAR Sequence of Processing

3 EXPERIMENTAL ANALYSIS

The scenario used in our experiments, Basic Central Planning, features two opposing teams with similar configuration. Both teams have a HQ agent and six subordinate Manoeuvre agents. The Blue team can send a series of messages that enable the HQ to make informed decisions and issue appropriate orders to subordinate agents. The opposing Red team, however, does not use communication between agents. Both teams are initialised on opposite sides of the map. Each agent in a team has 12 hit-points and starts the scenario with an initial order to defend their position for both Red HQ and Blue HQ agents and to advance to the opposing team's HQ location for all manoeuvre agents. In one scenario variation, red manoeuvre agents are instead ordered to defend their position. Our experiments are averaged over ten trials for each data point.

Blue HQ sends out orders to each Blue agent based on its state. A retreat order is given if an agent's hit-points fall to 50% or below. An attack order is issued when an enemy is detected and is sent out to all the agents irrespective of their current location, forcing the active blue agents to converge on the attack location.

Each Blue agent initially receives an order to advance toward the Red HQ location. This is their default behaviour, if no specific retreat or attack commands are received. When an agent receives a retreat order, it moves back towards the Blue HQ's location. Once an agent begins retreating, it will not accept any further orders. However, a retreating agent can still engage in combat if an enemy unit is encountered on its path to the Blue HQ.The Manouvre agents can not move while in combat. An incoming order is treated as new if it differs from the agent's current order.

The Red Team is initialised with the primary task of defending its spawn location. Red agents will engage Blue agents only if they are identified and enter the Red team's proximity of engagement. An agent's attack radius is 1 unit, while its detection range for enemy units is 2 units.

Table 3. Blue and Red Force % losses across four different message prioritisation policies.

Policy	Blue losses (%)	Red losess (%)
Static 1	76.38	69.66
Static 2 (En. pos.=pri. 1)	60.61	53.96
Dynamic 1	76.63	71.53
Dynamic 2	76.97	70.34

3.1 Effect of Rule Based Prioritisation on Mission Outcomes

We evaluate the effect of four different message type prioritisation strategies on mission outcomes as measured by the percentage of Blue and Red agents destroyed at the end of the run. Messages are sent from highest to

lowest priority, up to the allowed limit per agent per step:

Static 1: All message types have the same priority throughout the simulation run.
Static 2: One message type has a higher priority than all the others through out the simulation run. For the result in Table 3, Enemy position message type has the highest priority.
Dynamic 1: Order Update and Agent Deleted message types are assigned the highest priority. Enemy Location message type has medium priority, while the Status Update and Friendly Location message types have the lowest. However, when an agent enters combat, the prioritisation shifts dynamically with Status Update becoming a higher priority than Enemy Location, though they remain below Order Update and Agent Deleted message types.
Dynamic 2: All message types are assigned the same priority (Static 1 strategy). However, when an agent engages in combat, it adopts the message prioritisation protocol of the Dynamic 1 strategy.

Table 4. Effect of message bandwidth with no prioritisation.

Bandwidth	Blue loss (%)	Red loss (%)
None	58.27	58.34
Constrained	76.38	69.66
Unlimited	44.94	85.31

The dynamic strategies' focus on delivering Agent Deleted messages with the highest priority to ensure that no redundant actions are performed by HQ in response to out-of-date information, such as HQ ordering agents to move to agents that no longer exist. Similarly, our subject matter experts highlighted that when in combat, timely status updates are the most important as they enable retreat before an agent is destroyed. The results of these strategies, shown in Table 3, challenge this logic. For space constraint reasons, only the 'best' Static 2 strategy is shown. This strategy was not only the best Static 2 strategy, it also lead to lower blue force losses than either of the dynamic strategies. However, this strategy also lead to the lowest red force losses. Additionally the expert-designed dynamic strategies showed no significant advantage over Static 1 strategy. This highlights the difficulty in designing rule based strategies in complex environments and led to our exploration of RL techniques (discussed in Section 4). We also use SWAR to explore the effect of prioritisation in constrained bandwidth environments compared to no prioritisation, unconstrained communication and no communication. Table 4 shows a limited set of these results, showing that without prioritisation blue losses are higher in the constrained communication environments than in the no communication environments, with best results achieved with unlimited communication. This suggests that there are instances where incomplete information is worse than no information and highlights the importance of prioritisation.

4 Reinforcement Learning for Prioritisation

We have employed Reinforcement Learning (RL) to develop intelligent dissemination strategies. Unlike conventional RL applications that govern an agent's direct operational behaviour, such as movement or combat, our approach focuses on the message level. This allows agents to learn which messages are most important to send at any given moment, based on the current state and previous communications.

The SWAR environment facilitates the development of these dissemination strategies by allowing them to emerge from learned experience, guided by a reward signal. In the basic scenario used for RL experiments, rewards promote actions that damage adversaries while penalising friendly losses. Agents receive a reward of $+1$ for each enemy hit-point lost and -1 for each friendly hit-point lost.

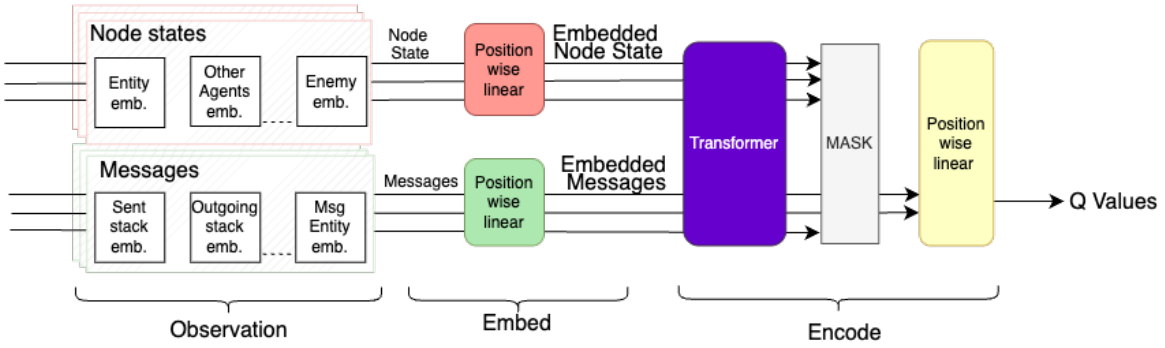

Figure 3. Reinforcement Learning Architecture

The observation space in SWAR consists of two components: a sequence of agent representations and a

sequence of message representations for each agent. The agent representations encapsulate the operational context of allies and detected enemies (e.g. hit-points, position, and combat/distress status). This representation is based on what is observed by an individual agent or received via messages. The message representation includes a sequence of received, outgoing, and sent messages, with attributes like message category and content. Every component of the agent representations and every message is provided along with a timestamp, allowing agents to assess whether the information is sufficiently up-to-date. Both the agent representation and a message representation is provided as a sequence of tokens as showcased in Figure 3.

The action space in SWAR allows agents to select any message for transmission from their outgoing message queue. As an agent progresses through an episode, this queue is populated with messages. The agent's action then involves choosing one message from this stack to send. This approach enables agents to learn and develop dynamic communication strategies instead of using static methods such as LIFO or FIFO

The SWAR Python Environment provides a Python wrapper for SWAR, suitable for RL that supports the PettingZoo (Terry et al. (2021)) multi-agent environment API. SWAR scenario/network parameters are also configurable via the Python interface. An episode within the SWAR Python Environment concludes when it reaches a predefined number of steps (500 in our experiments), or when all blue or all red units are defeated.

We employ SWAR to explore the use of several RL techniques, like, DQN, Independent Q Learning (IQL) (Kostrikov et al. (2021)) and QMIX. To manage long sequential message stacks, a Transformer architecture Vaswani et al. (2017) is integrated into our models. This design generates token representations for each observed blue and red agent, as well as for the various messages within their message stacks i.e. sent, outgoing, and received messages. The generation of message tokens includes a timestamp. This temporal data provides the RL policy with the necessary information to evaluate the reliability of the information. Since the agent and message tokens have different dimensions, they are first projected into a shared token space using a learned linear projection. Our policy architecture consists of a transformer encoder to jointly encode this sequence of agent and message tokens. This forms the backbone of our policy. The output from the encoder is then passed to a decoder, but we only focus on the tokens corresponding to outgoing messages. These tokens are decoded to predict the future cumulative reward (Q-value) of sending each message. Figure 3 represents our RL architecture. All Blue agents operate under a shared policy, a common practice in multi-agent RL known as parameter sharing (Li et al. (2024); Christianos et al. (2021)).

Table 5. Results from a simple SWAR Scenario. This scenario involves a 7 vs 7 configuration with no artillery agents. A send interval of 10 steps, with 2 messages sent at each interval. Messages had an expiry time of 20 steps, and the maximum outgoing message queue length was 10.

Policy	Rewards
No-op	-6.83
LIFO	6.62
FIFO	0.816
DQN (transformers)	15.25
QMIX (transformers)	12.544
IQL (transformers)	4.586

4.1 Effect of Learning on Mission Outcomes

In order to focus on Blue behaviour we use a modified scenario where Red agents are stationary while Blue agents are mobile. Each team comprises 7 units, including a HQ. The results of these experiments are presented in Table 5. We can see that unlike the rule based dynamic policies the DQN and QMIX algorithms significantly outperform the static policy. This indicates that learned communication strategies are more effective than static methods, demonstrating the value of SWAR as a tool for training RL policies for message dissemination. It is important to note that these are only preliminary results. Further experimentation and training is required in more complex scenarios to generate and validate RL based dissemination strategies that generalise across a wide range of mission contexts.

The learned policy exhibits conditional probabilities for sharing specific message types, 89.9% for status updates, 27.8% for enemy locations, and 10.3% for agent deletion messages. These probabilities were conditioned on the agent having a message of the corresponding message type available in the outgoing queue while selecting and sending the specific message type.

5 CONCLUSION

There is a critical need for simulation environments that facilitate the evaluation of the effectiveness of distributed decision making algorithms at the application level, where measures of effectiveness are difficult to define. This is particularly the case for decision making algorithms that operate at the network layer, where

performance metrics are easy to measure, and effectiveness metrics significantly more difficult. We propose SWAR as a simulation environment designed to explore the effect of different dissemination strategies on mission success. We demonstrated SWARs value in exploring the impact of rule based information dissemination strategies (message prioritisation) and shown that while the design of intelligent message prioritisation rules is critical for mission success, in a complex, adversarial system, even in relatively simple scenarios, this is a non trivial task. We have also described a framework for training RL policies in SWAR and shown that in limited scenarios DQN and QMIX with transformers can generate policies that outperform static strategies. Future work will focus on training RL algorithms on more complex strategies with the aim of generating policies for information dissemination that generalise across all expected mission contexts.

REFERENCES

Ahmad H., Treude C., Wagner M. and Szabo C. (2025), Towards resource-efficient reactive and proactive auto-scaling for microservice architectures, *Journal of Systems and Software* **225**, 112390.

Bernon C., Gleizes M, Peyruqueou S and Picard G (2003), Adelfe: A Methodology for Adaptive Multi-Agent Systems Engineering, *in* 'Engineering Societies in the Agents World III', pp. 156–169.

Boyapati Sree Ram and Szabo Claudia (2022), Self-adaptation in microservice architectures: A case study, *in* '2022 26th International Conference on Engineering of Complex Computer Systems', pp. 42–51.

Christianos F., Papoudakis G., Rahman M A and Albrecht S V (2021), Scaling multi-agent reinforcement learning with selective parameter sharing, *in* 'International Conference on Machine Learning', pp. 1989–1998.

Craggs D., Lee K. L., Radenovic V., Campbell B. and Szabo C. (2022), Use of reinforcement learning for prioritizing communications in contested and dynamic environments, *in* '2022 Winter Simulation Conference', pp. 2699–2711.

Eoyang G and Berkas T (1998), Evaluating performance in a cas, *Circle Pines, MN: Human Systems Dynamics Institute. Retrieved June* **21**, 2010.

Gopalan R., Shovon Md H.I., Campbell B., Radenovic V., McLeod K., Campbell L., Craggs D. and Szabo C. (2022), Message prioritization in contested and dynamic tactical networks using regression methods and mission context, *in* 'Winter Simulation Conference', pp. 2046–2057.

Jacyno Mariusz, Bullock Seth, Luck Michael and Payne Terry R (2009), Emergent service provisioning and demand estimation through self-organizing agent communities, *in* 'Proceedings of the International Conference on Autonomous Agents and Multiagent Systems-Volume 1', pp. 481–488.

Kostrikov Ilya, Nair Ashvin and Levine Sergey (2021), Offline reinforcement learning with implicit q-learning, *arXiv preprint arXiv:2110.06169* .

Kumar Mohit and Mishra Rashmi (2012), An overview of manet: History, challenges and applications, *Indian Journal of Computer Science and Engineering (IJCSE)* **3**(1), 121–125.

Li D., Lou N., Zhang B., Xu Z. and Fan G. (2024), Adaptive parameter sharing for multi-agent reinforcement learning, *in* 'IEEE International Conference on Acoustics, Speech and Signal Processing', pp. 6035–6039.

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D and Riedmiller M (2013), Playing atari with deep reinforcement learning, *arXiv preprint arXiv:1312.5602* .

Pumpuni-Lenss Gloria, Blackburn Timothy and Garstenauer Andreas (2017), Resilience in complex systems: an agent-based approach, *Systems Engineering* **20**(2), 158–172.

Rashid T., Samvelyan M. and De W. C. S et al. (2020), Monotonic value function factorisation for deep multi-agent reinforcement learning, *Journal of Machine Learning Research* **21**(178), 1–51.

Salazar Norman, Rodriguez-Aguilar Juan A, Arcos Josep Ll, Peleteiro Ana and Burguillo-Rial Juan C (2011), Emerging Cooperation on Complex Networks, *in* 'Proceedings of the International Conference on Autonomous Agents and Multiagent Systems', pp. 669–676.

Sproles Noel (2002), Formulating measures of effectiveness, *Systems Engineering* **5**(4), 253–263.

Suri N., Benincasa G., Lenzi R., Tortonesi M., Stefanelli C. and Sadler L. (2015), Exploring value of information-based approaches to support effective communications in tactical networks, *IEEE Communications Magazine* **53**(10), 39–45.

Szabo C., Craggs D., Balasoiu D.A., Radenovic V. and Campbell B. (2022), Robustness of middleware communication in contested and dynamic environments, *in* 'Winter Simulation Conference', pp. 2058–2069.

Terry J. K., Black B., Grammel N. and et al. M. Jayakumar (2021), Pettingzoo: Gym for multi-agent reinforcement learning.

Vaswani A., Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L and Polosukhin I (2017), Attention is all you need, *Advances in neural information processing systems* **30**.

