AFRL-RI-RS-TR-2023-049

REINFORCEMENT LEARNING AS A REHEARSAL FOR PLANNING IN AIR BATTLE MANAGEMENT (RLAR)

UNIVERSITY OF SOUTHERN MISSISSIPPI

MARCH 2023

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2023-049 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
SHAUN M. RYER, 1LT, USAF
Work Unit Manager

/ S /
JULIE BRICHACEK
Chief
Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
MARCH 2023	FINAL TECHNICAL REPORT	JUNE 2020	SEPTEMBER 2022

4. TITLE AND SUBTITLE

REINFORCEMENT LEARNING AS A REHEARSAL FOR PLANNING IN AIR BATTLE MANAGEMENT (RLAR)

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
N/A	FA8750-20-1-0105	62788F
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
		R310

6. AUTHOR(S)

Bikramjit Banerjee

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Southern Mississippi
2609 W 4th Street, Ste H
Hattiesburg MS 39401-5876

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Research Laboratory/RISB 525 Brooks Road Rome NY 13441-4505	AFRL/ RI	AFRL-RI-RS-TR-2023-049

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project leveraged some of the recent advances in RL to develop planners for real time strategy games, specifically MicroRTS in lieu of Stratagem program's wargame. One of these advances from the PI's lab is called reinforcement learning as a rehearsal (RLaR). Previously, RLaR had only been evaluated in toy benchmark tasks to establish its efficacy in sample complexity reduction. This project developed RLaR for the actor-critic architecture and applied it for the first time to a complex domain with incomplete information such as MicroRTS. Another technique applied in this project originated from the recent successes of multi-agent learning in the complex StarCraft II game, specifically the architecture of multi-stage training that develop league and league-exploiter policies during intermediate stages for training robust policies.

We trained RLaR against MicroPhantom—the runner-up from recent MicroRTS competitions—and showed its ability to plan effectively against this opponent but using fewer samples than relevant baselines. Separately, we trained RLaR in self-play using the 4-stage training scheme and evaluated the trained policy against MentalSeal (champion program) and MicroPhantom. While the policy once again showed good performance against MicroPhantom, it did not perform competently against MentalSeal. Based on an earlier preliminary finding that training against MentalSeal is extremely slow, we speculate that vastly more training time is required than what we could devote to this step during the extended period for this project.

15. SUBJECT TERMS

Artificial intelligence, machine learning, reinforcement learning, real-time strategy, turn-based strategy, wargames, Markov decision process, partially observable Markov decision process learning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	27

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
SHAUN M. RYER, 1LT, USAF	N/A

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 SUMMARY

This project leveraged some of the recent advances in RL to develop planners for real time strategy games, specifically MicroRTS in lieu of Stratagem program's wargame. One of these advances from the PI's lab is called reinforcement learning as a rehearsal (RLaR). Previously, RLaR had only been evaluated in toy benchmark tasks to establish its efficacy in sample complexity reduction. This project developed RLaR for the actor-critic architecture and applied it for the first time to a complex domain with incomplete information such as MicroRTS. Another technique applied in this project originated from the recent successes of multi-agent learning in the complex StarCraft II game, specifically the architecture of multi-stage training that develop league and league-exploiter policies during intermediate stages for training robust policies.

We trained RLaR against MicroPhantom—the runner-up from recent MicroRTS competitions—and showed its ability to plan effectively against this opponent but using fewer samples than relevant baselines. Separately, we trained RLaR in self-play using the 4-stage training scheme and evaluated the trained policy against MentalSeal (champion program) and MicroPhantom. While the policy once again showed good performance against MicroPhantom, it did not perform competently against MentalSeal. Based on an earlier preliminary finding that training against MentalSeal is extremely slow, we speculate that vastly more training time is required than what we could devote to this step during the extended period for this project.

2.0 INTRODUCTION

Although this project envisaged the application of reinforcement learning as a rehearsal (RLaR) to the Stratagem wargame, for various reasons it was restricted to a real time strategy (RTS) game called MicroRTS (Ontañón, 2013). RTS games belong to the genre of 2-player strategy games where a player's goal is to build sufficient economic and military might to destroy the opponent. A wide array of actions are available to a player, ranging from gathering resources, to building bases that train and churn out soldiers, to attacking opponent's units and bases to ultimately destroy them. For over two decades, RTS games have provided a rich substrate for AI research as they feature many of its key challenges, viz., complex dynamic environments with incomplete information and partial observability (fog-of-war), simultaneous and durative actions with potentially nondeterministic effects, real-time response, and unfathomably large strategy spaces. Consequently, this project's focus on the MicroRTS game was considered sufficient as it shares many of the same challenges with Stratagem.

Reinforcement learning (RL) has been a popular technique for training AI agents for computer games, including RTS games. Decades of research in this field boosted by the deep learning revolution have culminated in spectacular successes recently, where trained agents have matched and surpassed human expertise in domains where humans were once considered invulnerable to AI (Mnih, et al., 2015), (Vinyals, et al., 2019). However, RL remains a data-hungry approach that requires the agent to conduct a large number of simulations in order to comparatively evaluate a vast space of strategic alternatives. This is often measured as sample complexity. Despite decades worth of significant effort devoted toward reducing sample complexity, it still takes hundreds of millions of samples/simulations to train an RL agent in complex domains such as RTS games. In this project, we focus on a sample complexity reduction technique called reinforcement learning as a rehearsal (RLaR), and on the RTS game of MicroRTS to formulate and evaluate it. RLaR has been formulated in the context of action-value function based RL before (Kraemer & Banerjee, 2016). Here we formulate it for a different RL framework, called actor-critic RL. We show that on the one hand the actor-critic framework allows RLaR to be much simpler, but on the other hand it leaves room for a key component of RLaR–a prediction function that relates a learner's observations with that of its opponent. This function, when leveraged for exploration, accelerates RL as our experiments in MicroRTS show. Further experiments provide evidence that RLaR may reduce actor noise compared to a variant that does not utilize RLaR's exploration.

2.1 DETAILS OF MICRORTS

The components present in MicroRTS are bases, resources, barracks, worker units, and soldier units, as illustrated in Figure 1. A game is played between two players (learning agent controls the blue team), and the winner is determined when a player destroys all its opponent's units, including base, barracks and soldiers/units. If neither of the players is able to destroy its opponent's units within a given number of steps (3000 for this project), then it is a draw. Both players are given a worker unit, a base and 5 number of resources initially. Their locations, as well as the locations of unowned mineable resources, are symmetric to prevent either player from having an initial advantage. Worker units can harvest resources and build bases and barracks. Barracks produce soldier units of three types: light, heavy and ranged. Light units have less hitpoints whereas heavy units have high hitpoints, but both can only attack immediately neighboring cells. By contrast, ranged units can attack from 3 grid cells away. In this project, the learning agent is allowed to create up to $N_E = 70$ units–a number determined from game traces between MicroPhantom

(Richoux, 2020) and MentalSealPO—the top two players in the MicroRTS competition (Ontañón, Barriga, Silva, Moraes, & Lelis, 2018). More details about these champion programs can be found via https://sites.google.com/site/micrortsaicompetition. We also limit the map sizes to 16×16 in order to restrict training time. Actions available to a unit include "noop", "attack", and 4 directions each of "move", "harvest", "return", and "produce", leading to $N_A = 18$ action types. Actions "attack" and "produce" are further qualified by which location to attack and what type of unit to produce. Considering $N_T = 7$ types of units and up to 10 hitpoints, these choices lead to a state space of maximum size $(7 * 10)^{70+70} * \binom{256}{70+70} \approx 10^{333}$, assuming both players are allowed up to 70 units. The learner's observation space is of maximum size $(7 * 10)^{70} * \binom{128}{70} \approx 10^{166}$, assuming about half of the grid space is available to locate its units. Its action space is of maximum size $18^{70} \approx 10^{87}$, conservatively assuming only one attack location and one produce type per unit. This leads to a strategy (mapping from observations to actions) space that is truly unfathomable.

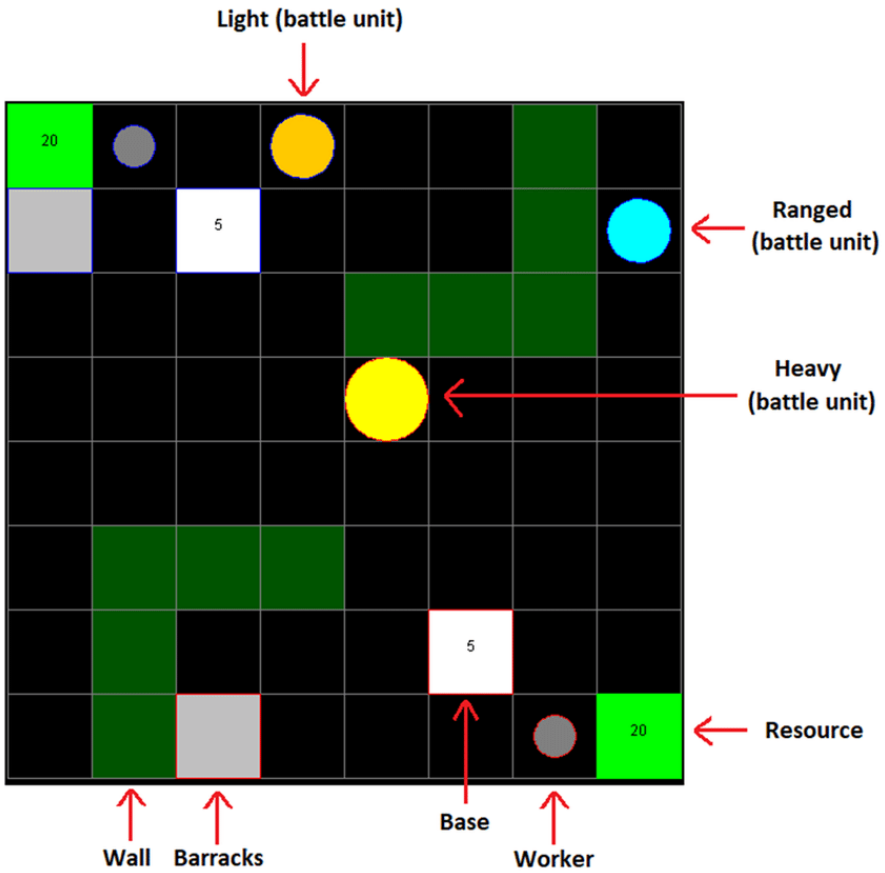


Figure 1: An 8X8 map of MicroRTS showing various kinds of units and features. This project focused on maps that are 16×16 or smaller.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 BACKGROUND

3.1.1 Reinforcement Learning

Reinforcement learning problems are modeled as Markov Decision Processes or MDPs (Sutton & Barto, 1998). An MDP is given by the tuple <S, A, R, P>, where S is the set of environmental states that an agent can occupy at any given time, A is the set of actions from which it can select one at a given state, $R : S \times A \rightarrow \Re$ is the reward function, i.e., R(s, a) specifies the reward from the environment that the agent gets for executing action $a \in A$ in state $s \in S$; $P : S \times A \times S \rightarrow [0, 1]$ is the state transition probability function, i.e., P(s, a, s') specifies the probability of the next state in the Markov chain being s' following the agent's selection of action a in state s. The agent's goal is to learn a policy $\pi : S \rightarrow A$ that maximizes the sum of current and future rewards from any state s, given by,

$$V^{\pi}(s_0) = E_P[\, R(s_0, \pi(s_0)) + \gamma.R(s_1, \pi(s_1)) + \gamma^2.R(s_2, \pi(s_2)) + \gamma^3.R(s_3, \pi(s_3)) + \dots] \quad \textbf{(1)}$$

where $s_0, s_1, s_2, \dots$ are successive samplings from the distribution P following the Markov chain with policy π, and $\gamma \in (0, 1)$ is a discount factor.

In this project we consider policy search methods (Sutton, McAllester, Singh, & Mansour, 2000) that explicitly maintain a policy $\pi_\theta(a|s)$ denoting the probability of taking action a in state s, with the distribution being parametrized by θ. In this project we use a policy gradient method—belonging to the class of policy search methods—where $\pi_\theta(a|s)$ is differentiable w.r.t θ.

One popular policy gradient technique, called Advantage Actor-Critic (A2C), uses two function approximations. One function approximation represents the actor, viz. $\pi_\theta(a|s)$ responsible for selecting an action given a state, as stated above. The other function approximation represents the critic, viz., $V^{\pi}_{\varphi}(s)$ which gives the value of the state s under the actor policy π (in essence it critiques the actor's performance), and is parametrized by φ. Normally θ is improved by policy gradient, optimizing

$$J(\theta) = E_{s \sim d^{\pi_\theta}, a \sim \pi_\theta} A(s, a) \quad \textbf{(2)}$$

where $d^{\pi_\theta}(s) = \sum_{t=0}^{\infty} \gamma^t Pr(s_t = s \mid s_0, \pi_\theta)$ is the discounted state distribution that results from following policy π_θ, and A(s, a) is called the advantage function that represents how much better (or worse) the value of taking action a in state s is compared to the average value from state s. A simple yet good estimate of the advantage function is the temporal difference (TD) error (Sutton, McAllester, Singh, & Mansour, 2000) given by

$$A_{TD}(s, a) = r_{sa} + \gamma.V^{\pi}_{\varphi}(s') - V^{\pi}_{\varphi}(s) \quad \textbf{(3)}$$

where $r_{sa} \sim R(s, a)$ and $s' \sim P(s, a, .)$. This estimate only depends on the reward and states from the actual trajectories and the critic itself. While the mean squared TD errors (from Eq. (3)) is used as the loss function for updating the parameters φ of the critic network, the actor network's parameters θ are updated using the gradient (Williams, 1992)

$$\nabla_\theta J(\theta) = E_{s\sim d^{\pi_\theta}, a\sim\pi_\theta} \nabla_\theta log\ \pi_\theta(a|s) A_{TD}(s,a) \quad \textbf{(4)}$$

In order to encourage exploration, an exploration bonus is added to the objective J(θ) whereby the entropy of the policy π_θ is also maximized, precluding the policy from settling into deterministic actions that could foreclose exploration. This gives a more complete expression for θ update:

$$\nabla_\theta J(\theta) = E_{s\sim d^{\pi_\theta}} [\ E_{a\sim\pi_\theta} \nabla_\theta log\ \pi_\theta(a|s) A_{TD}(s,a)\ -\ \beta \nabla_\theta \sum\nolimits_a \pi_\theta(a|s).\, log\ \pi_\theta(a|s)\ ] \quad \textbf{(5)}$$

where β is the entropy bonus (regularization) weight.

When the MDP is partially observable (POMDP), the state is not directly observed. Instead, the agent receives an observation, ω, that is (perhaps noisily) correlated with the hidden state. A common technique is to simply replace the states in the above equations with observations, or a history of past observations, as a sufficient statistic for the hidden state. In training neural networks π_θ and V^π_φ, history is accommodated via recurrence, e.g., using LSTM (Hochreiter & Schmidhuber, 1997). In this project, we use a variation of A2C, called A2C with self-imitation learning (A2C+SIL) (Oh, Guo, Singh, & Lee, 2018), where apart from the A2C loss functions a SIL loss function is added where advantages corresponding only to positive experiences are used. In other words, states where advantages are negative are zeroed out, thus simulating a learner's desire to recreate positive experiences from its past. This approach has been shown to be effective for hard exploration tasks.

3.1.2 Reinforcement Learning as a Rehearsal (RLaR)

RLaR (Kraemer & Banerjee, 2016) was designed for partially observable settings where a training stage could be distinguished from an execution stage where the learned policy is applied/evaluated. Furthermore, it was formulated in context of Q-learning (Watkins & Dayan, 1992), where an action-value function called Q-function is learned. It is related to the value function as follows:

$$V^\pi(\omega) = \max_a Q^\pi(\omega, a).$$

$Q^\pi(\omega, a)$ represents the long term value from following action a upon receiving observation ω, and the policy π thereafter. A Q-learning agent learns the optimal Q-values, $Q^*(\omega, a)\ \forall\omega, a$, and then constructs the optimal policy

$$\pi^*(\omega) = \arg\max_a Q^*(\omega, a).$$

RLaR allows a learner to observe the hidden state (s that includes system state as well as opponent's observations and actions) in addition to its observation (ω), but only during the training stage as if to practice/rehearse. A RLaR agent learns an augmented Qfunction, $Q^*(s, \omega, a)$, as well as an auxiliary predictor function (essentially a conditional probability distribution) $P(s|\omega)$, during the training/rehearsal stage. During the execution stage, the agent can construct a policy that no longer relies on hidden features, as

$$\pi^*(\omega)\ =\ arg\ max_a \sum_s Q^*(s,\omega,a) P(s|\omega).$$

This approach has been shown to expedite RL in simple 2-agent tasks (Kraemer & Banerjee, 2016), as well as in a larger swarm foraging task (Nguyen & Banerjee, 2022) more recently. In this project, we formulate RLaR within the actor-critic framework instead of Q-learning, and evaluate its effectiveness in a game with a large strategy space viz., MicroRTS.

3.2 METHODOLOGY

We develop the actor-critic architecture for RLaR using deep neural networks. The architectures of these networks are described next. Despite the existence of an OpenAI Gym framework (Huang, Ontañón, Bamford, & Grela, 2021) for RL in MicroRTS, we develop our own framework to gain the ability to (a) pass the hidden state to the RLaR agent, and (b) select any opponent of our choice (specifically, MicroPhantom and MentalSeal) instead of the fixed set of simpler opponents in (Huang, Ontañón, Bamford, & Grela, 2021).

3.2.1 The Actor Network

The architecture of the actor network, π_θ, is shown in Figure 2, and is used for all versions of RL studied here. Its input is the learner's observation at step t, ω_t, consisting of the following components

Scalar Features: Binary encoding of scalar features, e.g., time, score, resource;

Own Entities: Sparse encoding of its own units (their types, locations, health and resource);

Other Entities: Similar sparse encoding of other visible units either owned by the opponent, or unowned (e.g., harvestable resources);

Map: A 16×16 grid encoding of all visible units with their types.

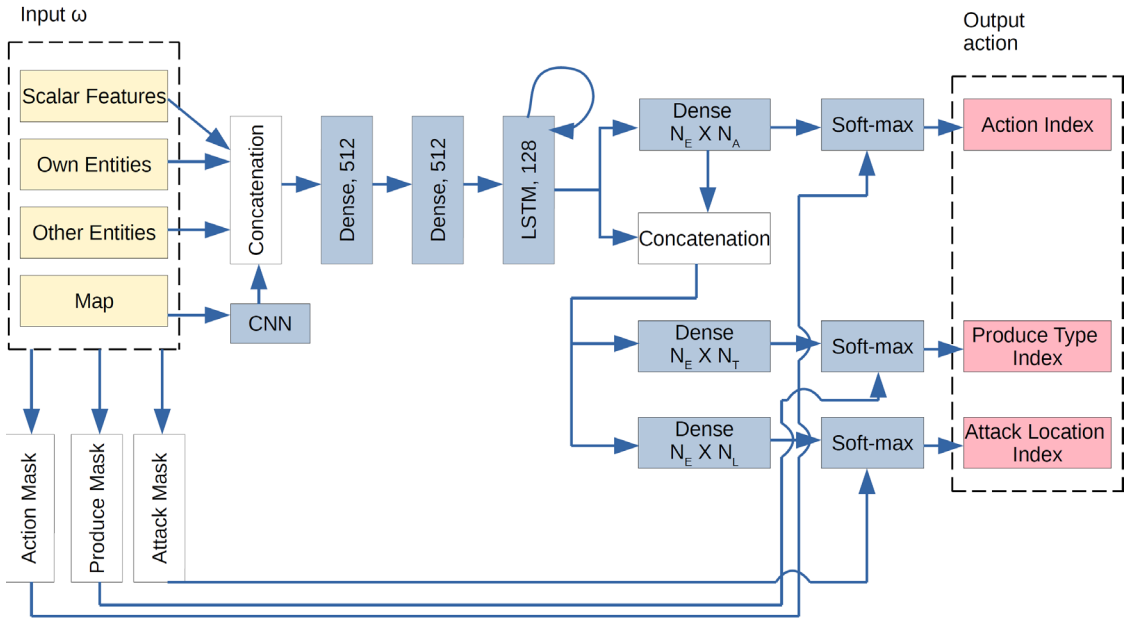


Figure 2: The actor network architecture.

The actor's output specifies the learner's action at step t, a_t. This is sampled from 3 soft-max probability distributions to yield the following:

> **Action Index**: For each of up to N_E (=70 in our experiments) units that the learner owns, one of (N_A=) 18 indices that encode noop, attack, and 4 directions each of move, harvest, return, and produce;
>
> **Produce Type Index**: If the produce action is selected for any of up to N_E units, the type index (from a set of N_T = 7 possible types) of what that unit will produce;
>
> **Attack Location Index**: If the attack action is selected for any of up to N_E units, the target location of the attack from a set of N_L (= 16^2 = 256) possible locations.

The soft-max layers are also provided with masks that reduce the support of the distributions, by deactivating elements that are invalid. Examples include movement directions that are blocked/occupied, harvest directions that do not contain resources, return directions that do not contain any self-base, produce types that are disallowed or require more resources than the agent/unit possesses, attack locations that are invisible or do not contain opponent units, etc. These masks allow the distributions to be learned rapidly despite the large strategy space, and are computable from ω_t and the information available from the unit type table provided at the beginning of the game. Similar invalid action masks are also used in (Huang, Ontañón, Bamford, & Grela, 2021).

3.2.2 The Critic Network

Let $s_t = (\omega_{1:t}, a_{1:t-1})$ be the observation-action history of the learner, and $s^-_t = (\omega^-_{1:t}, a^-_{1:t-1})$ be that of the opponent. Normally the opponent's observations are not available to a learner, hence for baseline RL the critic network learns the function $V_\varphi(s_t)$ as described in Section 3.1.1. A distinct feature of RLaR is that both the learner and opponent's observations are available to the learner during the training stage, and accommodated in its critic, $V^\pi_\varphi(s_t, s^-_t)$. Following (Kraemer & Banerjee, 2016), s^-_t can be marginalized out to compute a policy as

$$\pi * = \ arg\ max_\pi \sum_{s^-_t} P(s^-_t \mid s_t, \pi) V^\pi(s_t, s^-_t),$$

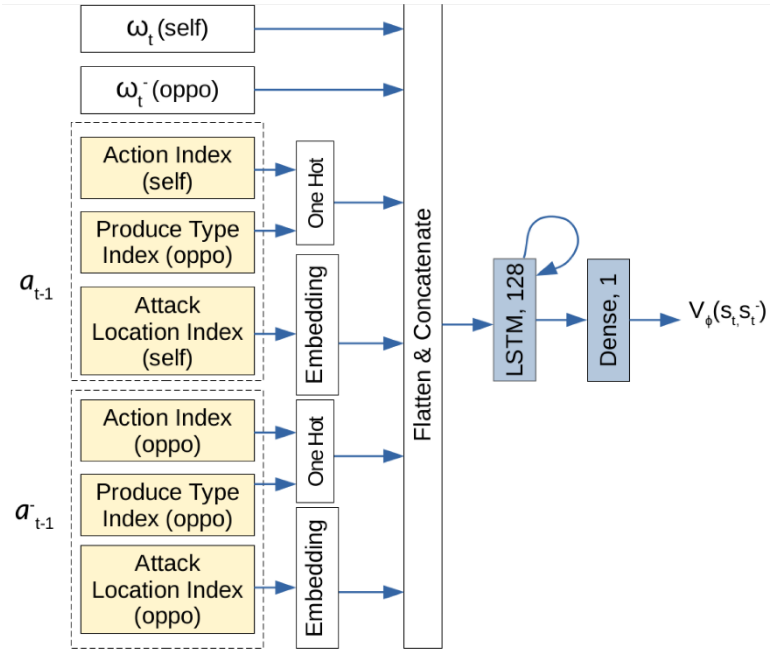


Figure 3: The critic network architecture.

using the learned auxiliary distribution $P(s_t^- \mid s_t, \pi)$. However, the actor-critic framework's clean separation of the policy from the value function makes this unnecessary. Since only the actor is needed after the training stage, and the critic is discarded, the accommodation of s^-_t in V is immaterial as long as the actor network is independent of s^-_t. Thus, for actor-critic training a simpler strategy is to exclude s^-_t altogether from the actor network, i.e., $\pi_\theta(a \mid s_t)$ instead of $\pi_\theta(a \mid s_t, s^-_t)$. This obviates the need for marginalization in the actor and allows us to use the actor network from Section 3.2.1 for all methods. Notice that s^-_t still impacts the actor updates since V is needed in equation (4) via equation (3). This strategy is followed in AlphaStar (Vinyals, et al., 2019), hence we call this approach RLAlpha and include it as a baseline in our experimental study. Both RLAlpha and RLaR use the critic network architecture shown in Figure 3. While N_A, N_T are small and are converted to one-hot representation, N_L is large and is therefore embedded. The critic for baseline RL simply omits ω^-_t and a^-_t in its input and is not shown separately. In contrast with the standard practice of combining the actor and critic networks to enable shared layers, we separate these networks such that the critics of the RL variants can be built incrementally without touching the actor.

3.2.3 Prediction Network for RLaR

Although the auxiliary distribution $P(s_t^- \mid s_t, \pi)$ was shown to be unnecessary for actor-critic in Sec. 3.2.2, there are still good reasons to learn it. An important feature of RLaR (as explained in (Kraemer & Banerjee, 2016)) is a principled incentive for exploration,

$$\pi_{explore} = arg\ min_\pi\ -\sum_{s_t^-} P(s_t^- \mid s_t, \pi)\ log\ P(s_t^- \mid s_t, \pi) \quad \textbf{(6)}$$

that seeks to reduce the entropy of the prediction $P(s_t^- \mid s_t, \pi)$. Ideally, if $P(s_t^- \mid s_t, \pi)$ is 1 then s_t is perfectly predictive of s^-_t under the current policy π, and the RLaR agent is truly independent of s^-_t. While RLAlpha does not have any incentive for this exploration, we can still endow RLaR with this capability for the following potential benefits:

- $\pi_{explore}$ may reduce noise in actor updates. Consider two situations where the learner observes s_t in both, but the opponent observes $s^-_{t,1}$ in one, and $s^-_{t,2}$ in another. While the critic can distinguish these situations being privy to $s^-_{t,1}$ and $s^-_{t,2}$, the actor cannot. If $V^\pi_\varphi(s_t, s^-_{t,1}) \neq V^\pi_\varphi(s_t, s^-_{t,2})$, then the resulting updates will appear as noise to the actor. However, if $P(s_t^- \mid s_t, \pi) = 1$ then $(s_t, s^-_t) \equiv s_t$ under π, and the above situation will not

materialize. Thus, $\pi_{explore}$ may push the actor toward generating situations where the updates are more stable.

- In the context of MicroRTS (and RTS games in general), $\pi_{explore}$ may encourage spying. In the partially observable setting of MicroRTS, a player can observe the set union of what its units can observe depending on their locations. Therefore, with strategically located units (a.k.a spies), a learner could make $\omega^{-}_{t} \subset \omega_t$, which would also minimize the entropy of $P(\omega_t^- \mid \omega_t, \pi)$. While spying may not be a worthwhile goal in and of itself, choosing actions with the knowledge of the opponent's configuration may be more desirable than without. Specifically, the success of the learned policy may be less dependent on the opponent's strategy, and more robust against other strategies.

Consequently, we seek to minimize the entropy of the distribution $P(\omega_t^- \mid \omega_{1:t}, \pi_{1:t-1})$, which reflects the objective of equation (6) more closely than P(s − t |st, π) in the context of MicroRTS. In particular, the condition $(\omega_{1:t}, \pi_{1:t-1})$ subsumes (s_t, π) as the action history embedded in s_t is sampled from the policy history $\pi_{1:t-1}$. Although MicroRTS allows the opponent's actions a^{-}_{t-1} to be observed partly/wholly as a part of ω_t with sufficient proximity, we focus on the prediction of ω^{-}_{t} alone, rather than s^{-}_{t} in order to restrict the size of the prediction network. To capture the conditional distribution $P(\omega_t^- \mid \omega_{1:t}, \pi_{1:t-1})$, we use a probabilistic auto-encoder (shown in Figure 4) similar to (Sohn, Lee, & Yan, 2015), albeit with an additional objective. In particular, an encoder network learns a latent representation of ω^{-}_{t} notated by latent variable Z, thus capturing the distribution $P(Z \mid \omega_t^-, \omega_{1:t}, \pi_{1:t-1})$. A decoder network is then tasked with reconstructing ω^{-}_{t} given inputs Z and $\omega_{1:t}$, $\pi_{1:t-1}$, thus inferring the distribution $P(\omega_t^- \mid Z, \omega_{1:t}, \pi_{1:t-1})$. Unlike (Sohn, Lee, & Yan, 2015), we do not use this auto-encoder as a generative model; yet we perform standard optimization of the variational evidence lower bound (ELBO) by minimizing the latent and reconstruction losses to update the predictor network, since it allows the latent variables to be distributed as $P(Z \mid \omega_{1:t}, \pi_{1:t-1})$. Our objective, in addition to the ELBO, is to minimize the entropy of this distribution. In order to serve as the exploration component (equation (6)), the gradients resulting from this entropy loss are only used to update the actor network, not the predictor network itself. The predictor update is solely based on the ELBO.

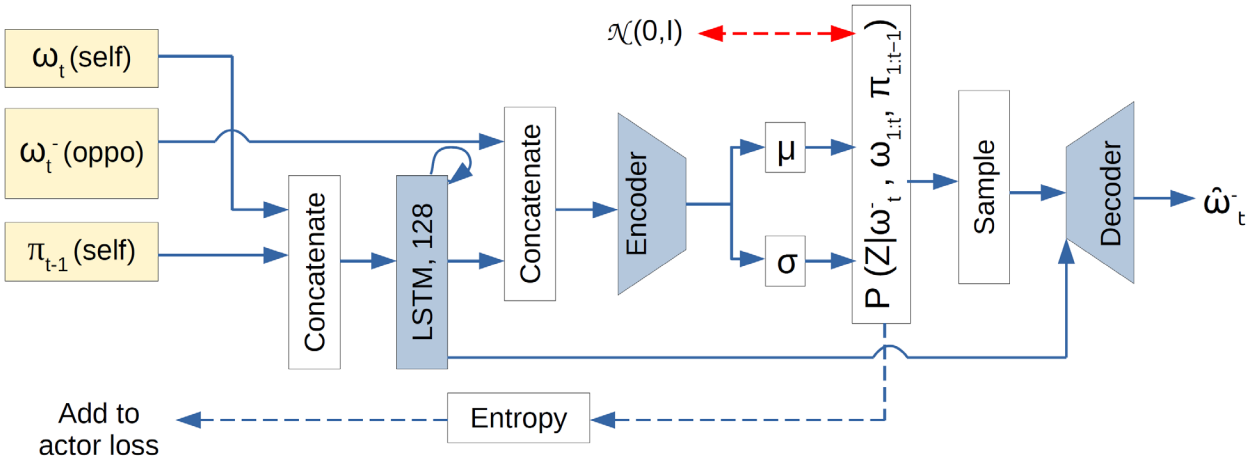


Figure 4: The prediction network architecture.

3.2.4 Four Stage Training Framework

As another important component of this project, we create a four-stage training framework for training red and blue agents using self-play, and evaluate their performance against the champion programs, MentalSeal and MicroPhantom, that are not used for training. The four stages are described next and illustrated in Figure 5.

Stage-I: In this stage, two RLaR agents are trained against each other, and the learned policies are saved when their performance against the other exceeds a certain threshold. These saved policies, called *league* policies, form two pools: the red league and the blue league. League policies tend to master certain skills but are usually not robust against a wide range of strategies.

Stage-II: In this stage, the red (blue) agent is paired (randomly in different episodes) with policies from blue (red) league, and the learned policies are saved periodically when the performance against the league opponent exceeds a certain threshold. This creates two new pools of policies that are explicitly optimized to defeat the league policies created in Stage-II, and are called red/blue *league exploiters*.

Stage-III: In this stage, the league exploiters created in the previous stage are made robust by further training them against many random pairings from league policies. This allows the league exploiters to master counter-skills to a wide range of league policy skills.

Stage-IV: In this final stage, a red (blue) RLaR agent is trained against the combined pool of red league exploiters (blue league exploiters) and blue (red) league policies. In each episode, a policy is randomly selected from this combined pool and paired against the RLaR agent. The outputs of this stage are the final red and blue agents that have been trained against a range of different opponents with various skills and counter-skills.

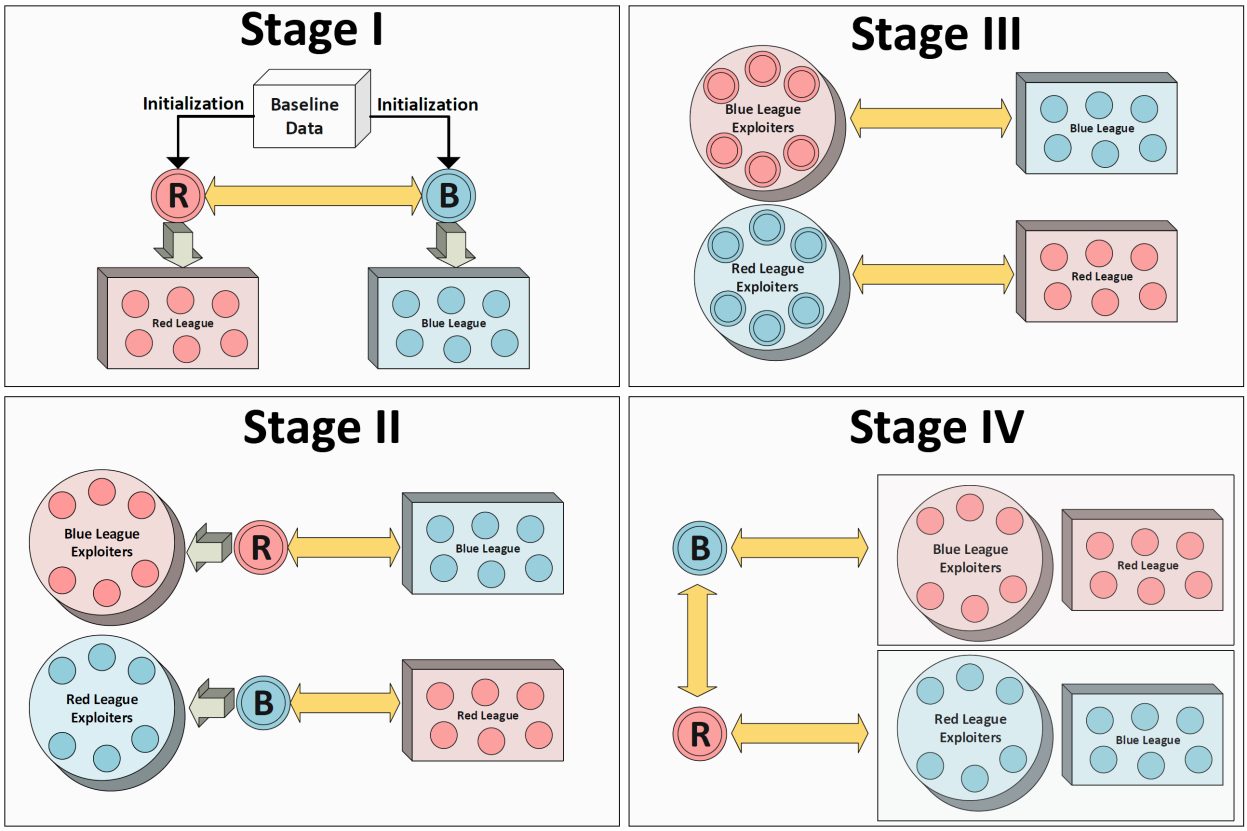


Figure 5: The four-stage training scheme.

4.0 RESULTS AND DISCUSSION

We conduct two sets of experiments to verify the efficacy of (1) RLaR and (2) the four-stage training framework in MicroRTS.

4.1 EVALUATION OF RLaR

We experiment with the three methods discussed in Section 3, viz., baseline RL, RLAlpha, and RLaR. For baseline RL, we use the advantage actor-critic (A2C) algorithm described in Section 3.1.1, modified with self-imitation learning (Oh, Guo, Singh, & Lee, 2018), A2C+SIL. Both RLAlpha and RLaR are built on top of A2C+SIL, thus sharing this common baseline. We train each variant in four different maps, shown in Figure 6 and Figure 7.

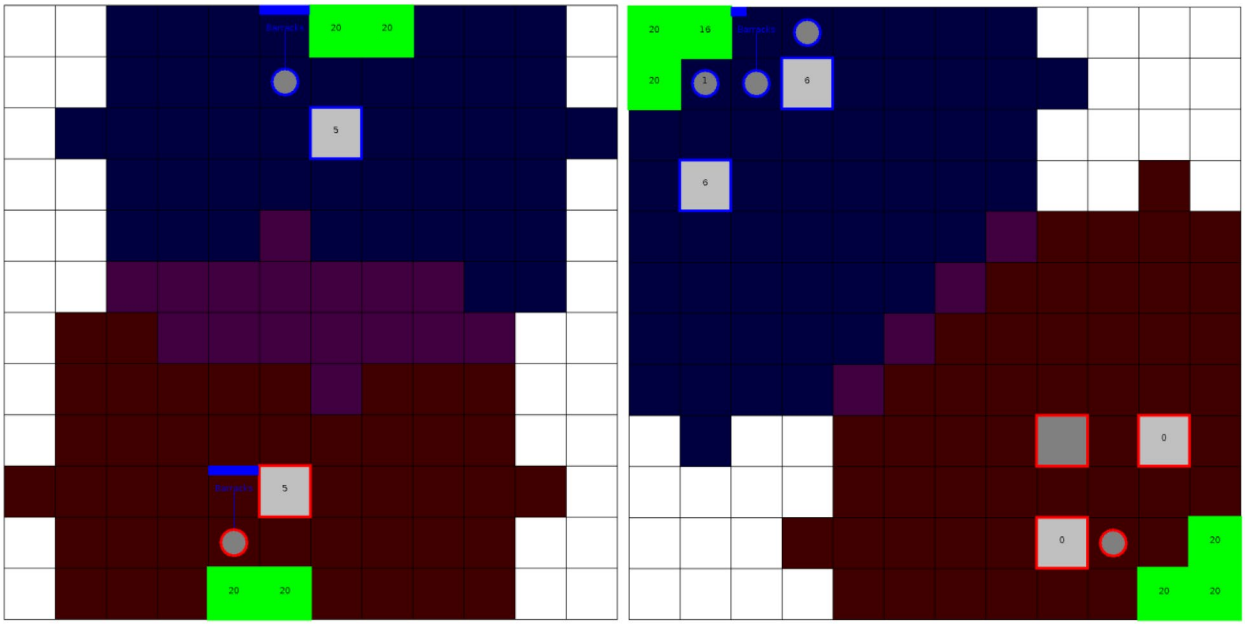


Figure 6: Maps "basesWorkers12X12F" on left, "complexBasesWorkers12X12" on right. White cells are unobserved, purple cells are observed by both blue and red teams. The learning agents always assume the role of the blue team, but there is no advantage to either role due to initial symmetry. The red team is MicroPhantom.

We selected these maps to incorporate variety of difficulty. For instance, the map "basesWorkers12x12F" (Figure 6 left) has the resources (bright green cells) in (relatively) opposite and non-corner locations, compared to other maps. The map "FourBasesWorkers12x12" (Figure 7 left) contains more initial bases and resources than other maps. Finally, the map "LetMeOut" (Figure 7 right) has a very different layout than other maps, where the players are walled (dark green cells) off, with doorways initially blocked by resources (although the blue agent had cleared one doorway by the time the screenshot was taken).

Games are capped at a maximum of 3000 steps. We use a sparse reward scheme, with 0 reward for any intermediate step, and non-zero rewards only for terminal steps: +1000 for a win, −1000 for a loss, 50+score for a draw (i.e., when a game does not complete within 3000 steps), where

score is the learner's MicroRTS assigned terminal score that reflects the strength/weakness of its final position in the absence of a clear winner. 50 bonus points are added for drawn games in order

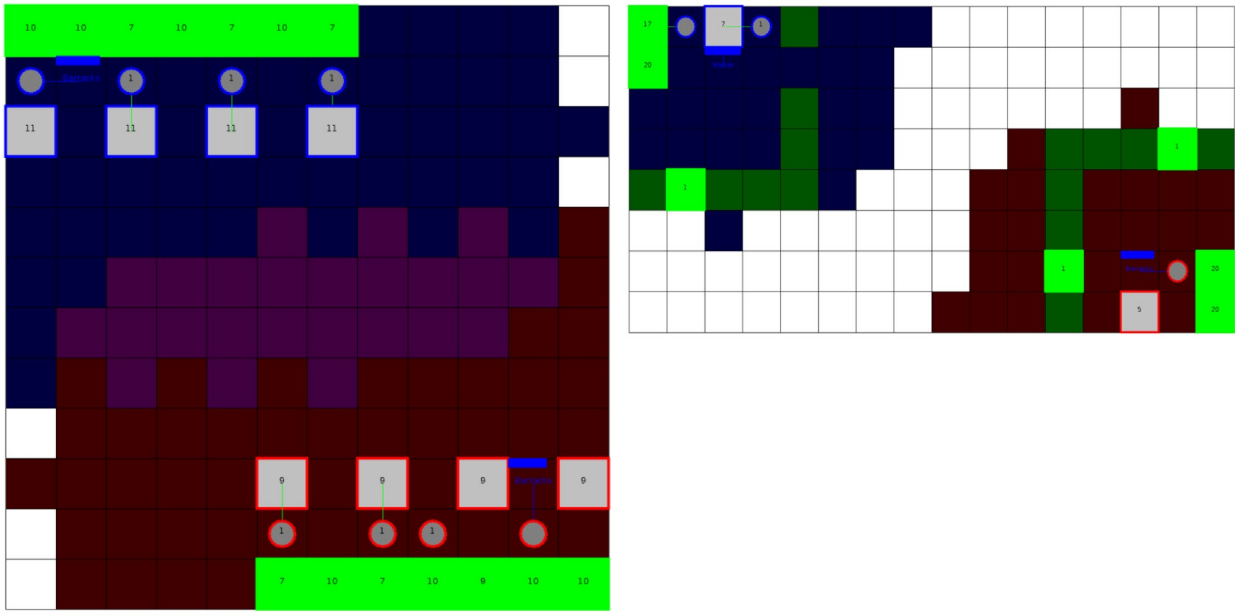


Figure 7: Maps "FourBasesWorkers12X12" on left, "LetMeOut" on right.

to avoid 0 returns for the entire trajectory when score = 0. The rest of the parameters are set as follows:

- $\gamma = 0.999$
- $\beta = 0.005$
- Actor learning rate = $5\times10-5$
- Critic learning rate = $5\times10-4$

The learning curves corresponding to the 4 maps are shown in Figure 8, over a series of 5500 games. Each curve is averaged over 6 independent trials, with half standard deviation bands shown in corresponding colors. The initial policy/actor for all versions were trained by supervised learning from a set of games played between MicroPhantom and MentalSeal. This results in positive initial performance of all variants, as seen in Figure 8 (b-d), although the trained initial policy was practically useless in (a). The learning curves demonstrate a superior learning rate for RLaR, and also serve as an ablation for the predictor network as that is the only difference between RLAlpha and RLaR. Also note that a total reward approaching +1000 indicates that the agent has learned to almost always defeat MicroPhantom. Videos of trained RLaR policy against MicroPhantom are posted at https://tinyurl.com/y3xhb9nt. Baseline RL is not shown in Figure 8 as its performance is poor in comparison with RLAlpha and RLaR. In particular, starting with the trained initial policy, baseline RL essentially unlearns it, dropping the total reward to -1000 (even in maps (b-d)) before improving it again. Essentially, baseline RL is unable to leverage the initial policy at all, requiring more time to learn. We show the performance of the learned policy at the end of 5500

games for all three variants in Table 1. Table 1 clearly demonstrates the futility of single agent (baseline A2C+SIL) RL in the face of a large strategy space. Although the centralized (i.e., joint)

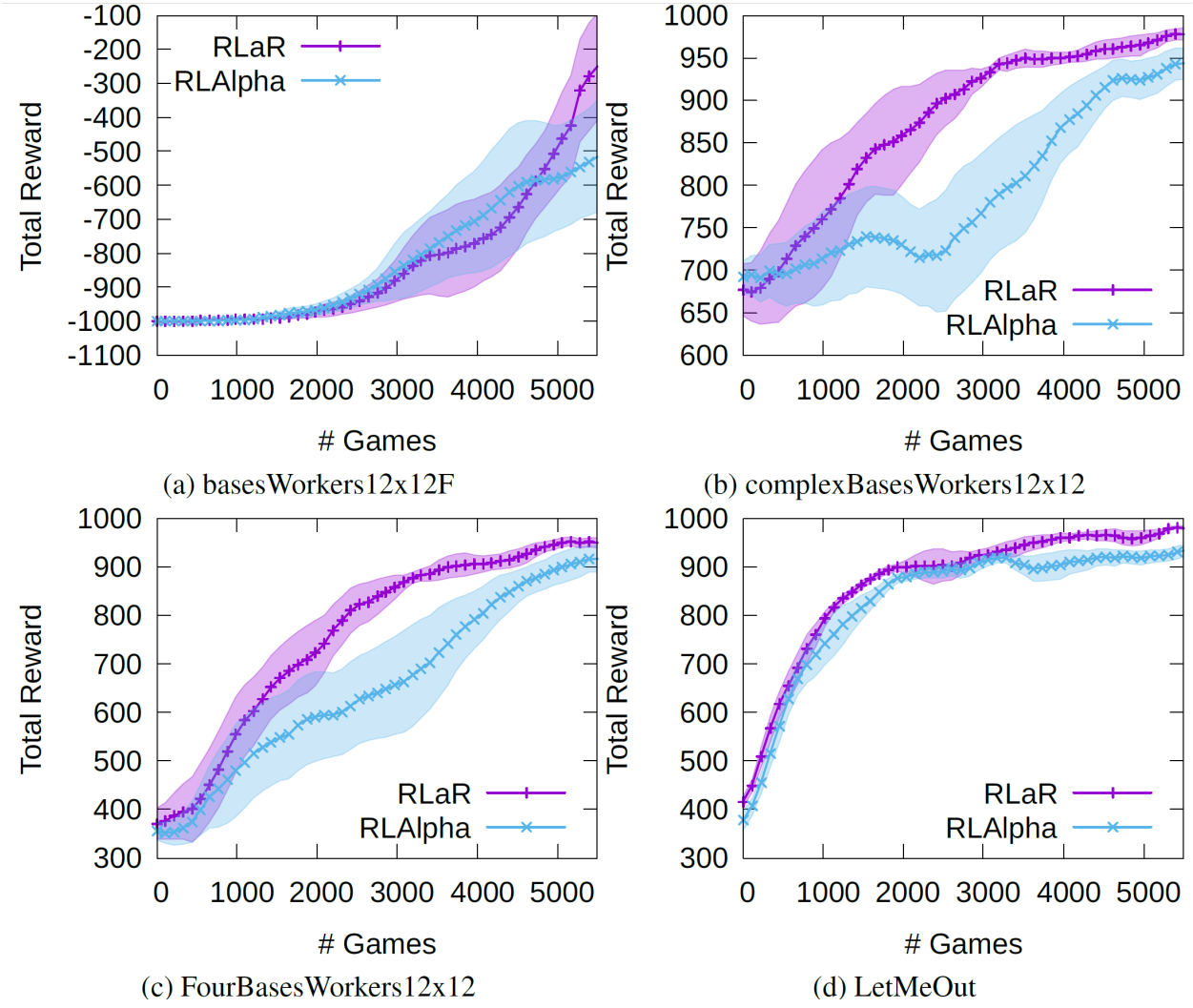


Figure 8: Learning curves of RLAlpha (RLAlpha+A2C+SIL) and RLaR RLaR+A2C+SIL) against MicroPhantom in the 4 maps of Figures 6, 7. Baseline RL (A2C+SIL) is excluded due to poor performance. The terminal rewards for win/loss/draw are +1000/-1000/+50. The initial policy/actor was trained by supervised learning from games between MentalSeal and MicroPhantom on large set of maps, but performs poorly in (a).

critic of RLAlpha brings it closer to RLaR, Table 1 also demonstrates the scope for further improvement in terms of a principled exploration component that is unique to RLaR.

Table 1: Performance of trained policies for 3 variants in 4 maps.

Maps	RL(A2C+SIL)	RLAlpha	RLaR
basesWorkers12X12F	-998.7 ± 2.0	-515.9 ± 164.9	**-249.7 ± 161.0**
complexBasesWorkers12X12	591.9 ± 73.1	943.6 ± 18.5	**977.8 ± 8.9**
FourBasesWorkers12X12	254.8 ± 118.9	916.5 ± 26.4	**949.4 ± 11.5**
LetMeOut	125.5 ± 290.7	933 ± 12.5	**979.6 ± 4.1**

In order to further evaluate the impact of RLaR's characteristic exploration, we conduct a second experiment. In this experiment, we note the number of steps in a game that it takes the learner to get close enough to the opponent's base, i.e., for any of its units to get within a distance threshold of the opponent's base. When there are multiple opponent bases, we take the centroid of their locations. This can be viewed as a rough measure of how quickly the learner deploys spies.

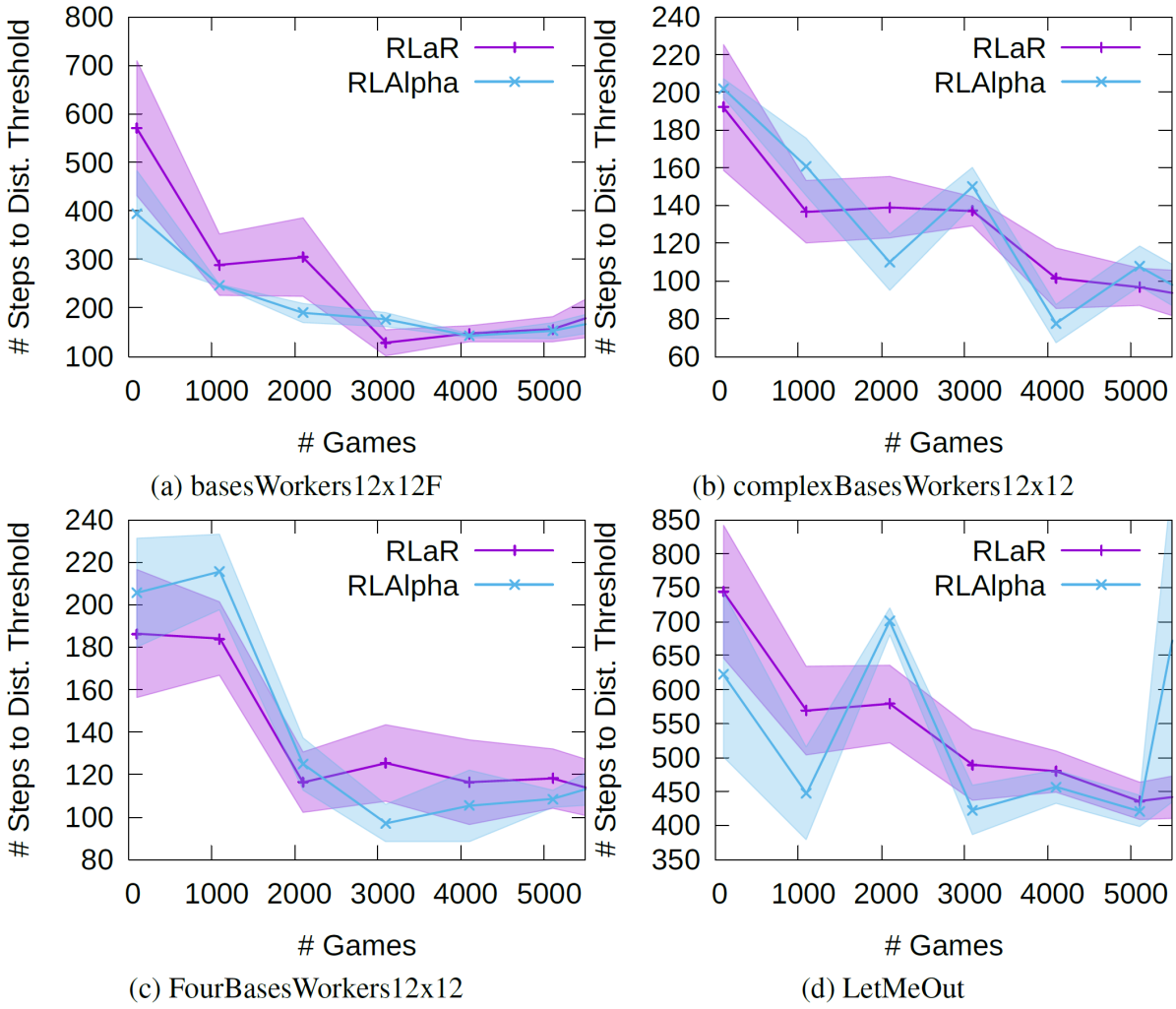
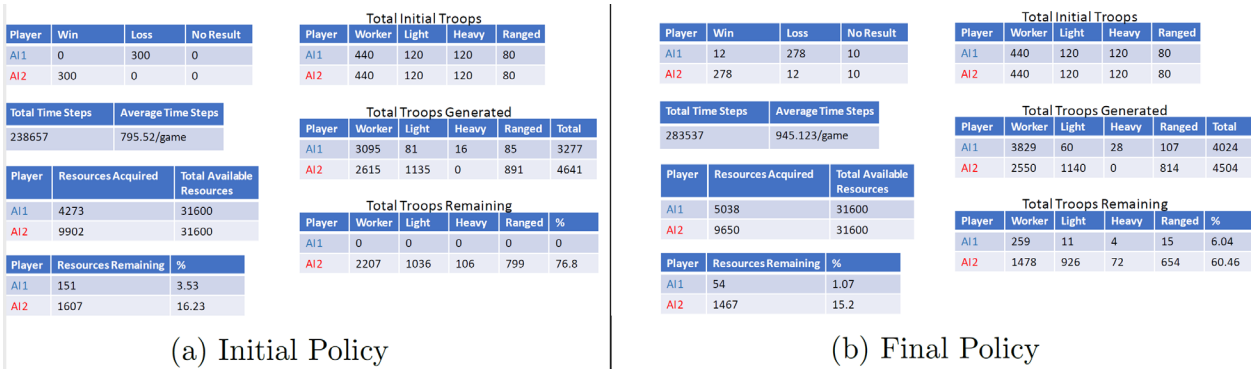


Figure 9: Plots showing the number of steps that the learner needs before at least one of its units gets within a distance threshold of 4.0 of the opponent's base, thereby bringing it within the radius of the learner's visibility.

The results are shown in Figure 9 for a distance threshold of 4.0–sufficient to bring it within the observable radius. The first observation is that this measure does not correlate accurately with learning performance (Figure 8), as early spying can end in failure while late spying can still end in victory. Neither is it a measure of the effectiveness of spying, as observing the opponent's base does not mean all of the opponent's units are also visible. However, another observation from Figure 9 is that while the trend is expected to be decreasing with continued learning, this does not occur reliably with RLAlpha. Particularly in Figure 9 (b) and (d), we notice spikes where the learner appears to be regressing in terms of this measure. RLaR, by contrast, achieves a steadier acceleration toward proximity. As proximity is a reliable predictor of the opponent's observation in MicroRTS, we speculate that this is a direct result of RLaR's use of predictor based exploration.

4.2 EVALUATION OF 4-STAGE LEARNING FRAMEWORK

In Figures 10--13, we show the comparative performances of RLaR's initial policy and the policy from the 4-th/final stage, against MentalSeal and MicroPhantom, playing either roles 0 (blue) or 1 (red).



Player	Win	Loss	No Result
AI1	0	300	0
AI2	300	0	0

Total Time Steps	Average Time Steps
238657	795.52/game

Player	Resources Acquired	Total Available Resources
AI1	4273	31600
AI2	9902	31600

Player	Resources Remaining	%
AI1	151	3.53
AI2	1607	16.23

Total Initial Troops

Player	Worker	Light	Heavy	Ranged
AI1	440	120	120	80
AI2	440	120	120	80

Total Troops Generated

Player	Worker	Light	Heavy	Ranged	Total
AI1	3095	81	16	85	3277
AI2	2615	1135	0	891	4641

Total Troops Remaining

Player	Worker	Light	Heavy	Ranged	%
AI1	0	0	0	0	0
AI2	2207	1036	106	799	76.8

(a) Initial Policy

Player	Win	Loss	No Result
AI1	12	278	10
AI2	278	12	10

Total Time Steps	Average Time Steps
283537	945.123/game

Player	Resources Acquired	Total Available Resources
AI1	5038	31600
AI2	9650	31600

Player	Resources Remaining	%
AI1	54	1.07
AI2	1467	15.2

Total Initial Troops

Player	Worker	Light	Heavy	Ranged
AI1	440	120	120	80
AI2	440	120	120	80

Total Troops Generated

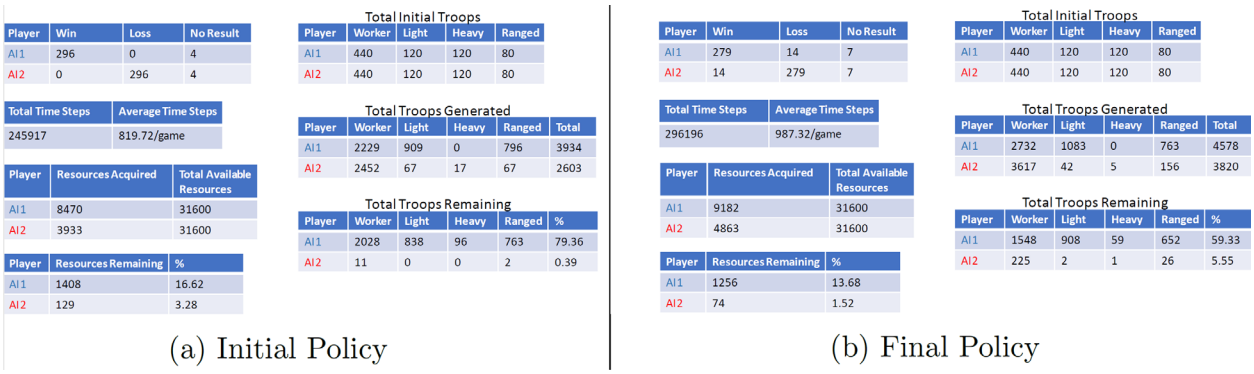
Player	Worker	Light	Heavy	Ranged	Total
AI1	3829	60	28	107	4024
AI2	2550	1140	0	814	4504

Total Troops Remaining

Player	Worker	Light	Heavy	Ranged	%
AI1	259	11	4	15	6.04
AI2	1478	926	72	654	60.46

(b) Final Policy

Figure 10: Accumulated metrics of initial and final policies against MentalSeal playing as Red, in 30 maps (10 games/map, total 300 games). Here, RLaR is Blue.



Player	Win	Loss	No Result
AI1	296	0	4
AI2	0	296	4

Total Time Steps	Average Time Steps
245917	819.72/game

Player	Resources Acquired	Total Available Resources
AI1	8470	31600
AI2	3933	31600

Player	Resources Remaining	%
AI1	1408	16.62
AI2	129	3.28

Total Initial Troops

Player	Worker	Light	Heavy	Ranged
AI1	440	120	120	80
AI2	440	120	120	80

Total Troops Generated

Player	Worker	Light	Heavy	Ranged	Total
AI1	2229	909	0	796	3934
AI2	2452	67	17	67	2603

Total Troops Remaining

Player	Worker	Light	Heavy	Ranged	%
AI1	2028	838	96	763	79.36
AI2	11	0	0	2	0.39

(a) Initial Policy

Player	Win	Loss	No Result
AI1	279	14	7
AI2	14	279	7

Total Time Steps	Average Time Steps
296196	987.32/game

Player	Resources Acquired	Total Available Resources
AI1	9182	31600
AI2	4863	31600

Player	Resources Remaining	%
AI1	1256	13.68
AI2	74	1.52

Total Initial Troops

Player	Worker	Light	Heavy	Ranged
AI1	440	120	120	80
AI2	440	120	120	80

Total Troops Generated

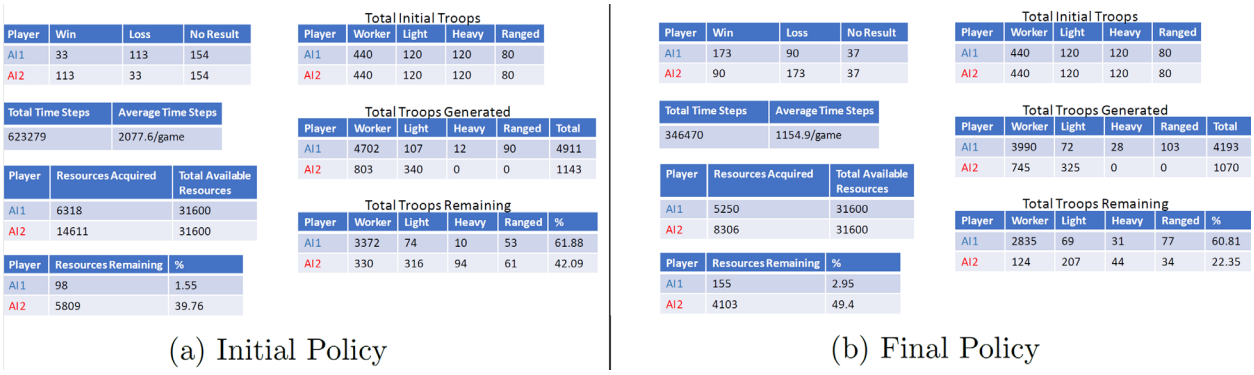
Player	Worker	Light	Heavy	Ranged	Total
AI1	2732	1083	0	763	4578
AI2	3617	42	5	156	3820

Total Troops Remaining

Player	Worker	Light	Heavy	Ranged	%
AI1	1548	908	59	652	59.33
AI2	225	2	1	26	5.55

(b) Final Policy

Figure 11: Accumulated metrics of initial and final policies against MentalSeal playing as Blue, in 30 maps (10 games/map, total 300 games). Here, RLaR is Red.



Player	Win	Loss	No Result
AI1	33	113	154
AI2	113	33	154

Total Time Steps	Average Time Steps
623279	2077.6/game

Player	Resources Acquired	Total Available Resources
AI1	6318	31600
AI2	14611	31600

Player	Resources Remaining	%
AI1	98	1.55
AI2	5809	39.76

Total Initial Troops

Player	Worker	Light	Heavy	Ranged
AI1	440	120	120	80
AI2	440	120	120	80

Total Troops Generated

Player	Worker	Light	Heavy	Ranged	Total
AI1	4702	107	12	90	4911
AI2	803	340	0	0	1143

Total Troops Remaining

Player	Worker	Light	Heavy	Ranged	%
AI1	3372	74	10	53	61.88
AI2	330	316	94	61	42.09

(a) Initial Policy

Player	Win	Loss	No Result
AI1	173	90	37
AI2	90	173	37

Total Time Steps	Average Time Steps
346470	1154.9/game

Player	Resources Acquired	Total Available Resources
AI1	5250	31600
AI2	8306	31600

Player	Resources Remaining	%
AI1	155	2.95
AI2	4103	49.4

Total Initial Troops

Player	Worker	Light	Heavy	Ranged
AI1	440	120	120	80
AI2	440	120	120	80

Total Troops Generated

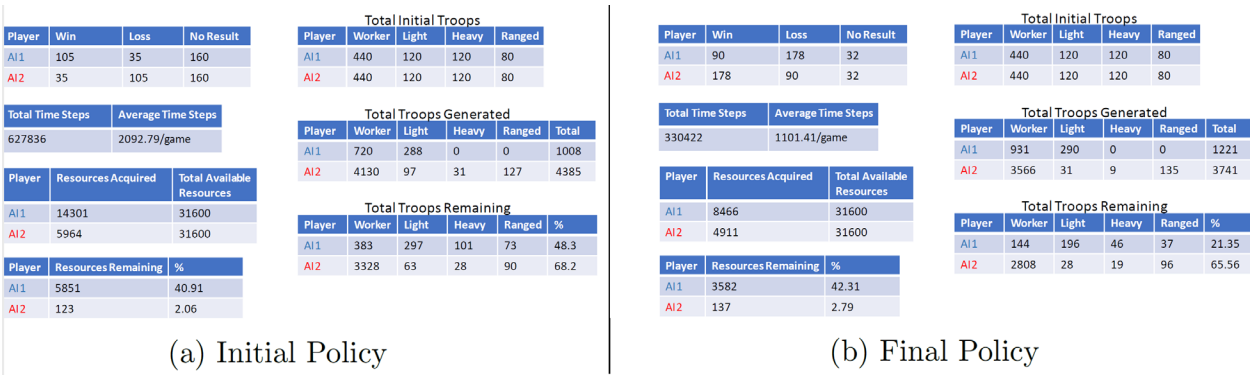
Player	Worker	Light	Heavy	Ranged	Total
AI1	3990	72	28	103	4193
AI2	745	325	0	0	1070

Total Troops Remaining

Player	Worker	Light	Heavy	Ranged	%
AI1	2835	69	31	77	60.81
AI2	124	207	44	34	22.35

(b) Final Policy

Figure 12: Accumulated metrics of initial and final policies against MicroPhantom playing as Red, in 30 maps (10 games/map, total 300 games). Here, RLaR is Blue.



Player	Win	Loss	No Result
AI1	105	35	160
AI2	35	105	160

Total Time Steps	Average Time Steps
627836	2092.79/game

Player	Resources Acquired	Total Available Resources
AI1	14301	31600
AI2	5964	31600

Player	Resources Remaining	%
AI1	5851	40.91
AI2	123	2.06

Total Initial Troops

Player	Worker	Light	Heavy	Ranged
AI1	440	120	120	80
AI2	440	120	120	80

Total Troops Generated

Player	Worker	Light	Heavy	Ranged	Total
AI1	720	288	0	0	1008
AI2	4130	97	31	127	4385

Total Troops Remaining

Player	Worker	Light	Heavy	Ranged	%
AI1	383	297	101	73	48.3
AI2	3328	63	28	90	68.2

(a) Initial Policy

Player	Win	Loss	No Result
AI1	90	178	32
AI2	178	90	32

Total Time Steps	Average Time Steps
330422	1101.41/game

Player	Resources Acquired	Total Available Resources
AI1	8466	31600
AI2	4911	31600

Player	Resources Remaining	%
AI1	3582	42.31
AI2	137	2.79

Total Initial Troops

Player	Worker	Light	Heavy	Ranged
AI1	440	120	120	80
AI2	440	120	120	80

Total Troops Generated

Player	Worker	Light	Heavy	Ranged	Total
AI1	931	290	0	0	1221
AI2	3566	31	9	135	3741

Total Troops Remaining

Player	Worker	Light	Heavy	Ranged	%
AI1	144	196	46	37	21.35
AI2	2808	28	19	96	65.56

(b) Final Policy

Figure 13: Accumulated metrics of initial and final policies against MicroPhantom playing as Red, in 30 maps (10 games/map, total 300 games). Here, RLaR is Red.

As the 4-stage training in Figure 5 indicates, the opponents MentalSeal or MicroPhantom were never used in training the policies at any stage. Despite this, the trained policy beat MicroPhantom in many games that were being drawn initially (Figure 12, Figure 13). However, the policy's performance against MentalSeal (Figure 10, Figure 11) remains unimpressive. In particular, the final policy wins 12-14 matches whereas the initial policy lost almost all matches.

We encountered a couple of challenges that may have precluded more impressive results w.r.t. MentalSeal. One was unforeseen, where the criterion for selecting league policies to be saved periodically in Stage-I did not fire evenly for Red and Blue. Instead, many more Blue League policies were collected (> 40) compared to only a limited number (8) of Red League policies. To mitigate the effect of this unevenness on subsequent stages, we had to ignore most of the Blue League policies and use the top 8 of the Blue League policies to match 8 Red League policies, in Stages II-IV. Another challenge was foreseeable from our general experience throughout the project: slow training. As a result, each of Stages I-III had to be limited to 1000-1500 matches in order to allow the GPUs to be used for testing and debugging of subsequent stages. We believe, with more training time in these earlier stages, the trained Red and Blue policies would have performed better against MentalSeal.

Although the neural network policy mines fewer resources compared to MentalSeal (about half as much), it utilizes almost all of it, more so in the trained version than the initial policy. By constrast, MentalSeal underutilizes a significant amount, $\approx 16\%$, of mined resources. This contrast is starker against MicroPhantom, which acquires even more resources than MentalSeal, but leaves a larger percentage underutilized, $> 40\%$. In terms of troop generation, the 4 stage training leads to the creation of more units of Ranged type compared to the initial policy, but not quite as many as MentalSeal. MentalSeal's utilization of a much larger number of Light and Ranged troops, compared to RLaR, may be the cornerstone of its superior performance. When compared against MicroPhantom this intuition is reinforced as it never produces any Ranged troops and performs poorly against the trained RLaR policy. However, it does produce more light troops than the RLaR policy.

5.0 CONCLUSIONS

We have developed a principled formulation of reinforcement learning as a rehearsal (RLaR) for the first time within the actor-critic framework. We have shown how a key component of RLaR, a prediction function that correlates the opponent's observations to the learner's own observations, can be constructed within a deep learning pipeline. Although the formulation is in the context of MicroRTS, it can be easily extended to other RTS games, e.g., StarCraft, and potentially to the Stratagem wargame. We have experimentally validated two of the benefits of RLaR compared to a variant that has all the same features as RLaR except the prediction function. Consistent with previous findings on RLaR in smaller strategy spaces, we have shown that RLaR improves learning speed even in a domain with a large strategy space such as MicroRTS. A second experiment has shown that RLaR achieves visibility of the opponent's base more predictably as learning progresses. We speculate that this might be indirect evidence of noise reduction in actor updates–a second benefit of our approach–and at least partly responsible for improved learning rate of RLaR.

Reward shaping (Ng, Harada, & Russell, 1999) is a well-established technique in RL where domain/prior knowledge is often used to supplement the reward function, in order to shape and accelerate learning. It is conceivable that a shaping function that rewards a learner for observing more of the opponent's units and penalizes it for observing less, could achieve similar learning speedup as RLaR in this project, because that is a known effect of reward shaping. Additionally, it might also achieve similar noise reduction, since the effect of such shaping on the actor in terms of the generated trajectories is likely to be similar. Further experiments can be conducted in the future to evaluate these intuitions. In contrast with this potentially alternative approach, we have relied on a simple (sparse) reward scheme in this project, and avoided explicit domain-specific reward engineering. More importantly, our approach is more general than reward shaping, as shaping functions can vary from domain to domain, but entropy minimization of the prediction function is a general principle that does not need domain-specific engineering, and can benefit domains well beyond RTS games.

We have also conducted the 4-stage training of RLaR policies and evaluated the initial and final policies against MentalSeal and MicroPhantom. Although these sophisticated opponents were not used for training in this experiment, the trained policy was able to beat MicroPhantom in a large set of games. However, its performance against MentalSeal remains unimpressive. As this final part of the project was conducted during the 3 month extension, and as the training was slow, we were unable to run the earlier stages (Stage I-III) long enough. This may have handicapped the final product, which is why we recommend that these stages be run for longer than the 1000-1500 episodes that we managed to run, possibly ~10,000 episodes each. Apart from raw number of wins/losses, we have also observed qualitative differences in how the RLaR policy utilizes resources and soldiers compared to MentalSeal and MicroPhantom.

A large part of this project was conducted as a major coding project with an eye toward participation in the MicroRTS competition. As a result, only one thesis and one paper resulted from it. Unfortunately, the competition was discontinued in 2022—the year we had targeted. In hindsight, had we not set out sights on the competition, we could have focused more on the science possibly leading to more publications.

6.0 REFERENCES

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*, 1735–1780.

Huang, S., Ontañón, S., Bamford, C., & Grela, L. (2021). {Gym-μRTS: Toward Affordable Full Game Real-time Strategy Games Research with Deep Reinforcement Learning}. *2021 IEEE Conference on Games (CoG).* Retrieved from https://arxiv.org/abs/2105.13807

Kraemer, L., & Banerjee, B. (2016). Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing, 190*, 82–94.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., . . . Hassabis, D. (2015, February). Human-level control through deep reinforcement learning. *Nature, 518*, 529-33. doi:10.1038/nature14236

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *Proc. 16th International Conf. on Machine Learning* (pp. 278–287). Morgan Kaufmann.

Nguyen, T., & Banerjee, B. (2022). Reinforcement Learning as a Rehearsal for Swarm Foraging. *Swarm Intelligence, 16*, 29–58.

Oh, J., Guo, Y., Singh, S., & Lee, H. (2018). Self-Imitation Learning. *ICML.*

Ontañón, S. (2013). The Combinatorial Multi-Armed Bandit Problem and its Application to Real-Time Strategy Games. *Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)* (pp. 58–64). Boston: AAAI.

Ontañón, S., Barriga, N. A., Silva, C. R., Moraes, R. O., & Lelis, L. H. (2018, March). The First microRTS Artificial Intelligence Competition. *AI Magazine, 39*, 75-83. doi:10.1609/aimag.v39i1.2777

Richoux, F. (2020). MicroPhantom: Playing MicroRTS under uncertainty and chaos. *2020 IEEE Conference on Games (CoG)*, (pp. 670-677). doi:10.1109/CoG47356.2020.9231653

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems, 28*.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems 12* (pp. 1057–1063). MIT Press.

Sutton, R., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press.

Vinyals, O., Babuschkin, I., Czarnecki, W., Mathieu, M., Dudzik, A., Chung, J., . . . Silver, D. (2019, November). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature, 575*. doi:10.1038/s41586-019-1724-z

Watkins, C., & Dayan, P. (1992). Q-learning. *Machine Learning, 3*, 279-292.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*, 229–256.

APPENDIX A – PUBLICATIONS AND PRESENTATIONS

Presentation

The PI participated in the "Autonomy, Command & Control PI Meeting" (AC2 AFRL PI Meeting) held virtually between Mar-15 and Mar-18, 2021. The PI presented the work done up to that point on Mar-15-2021 during the morning session. The title of the talk was also "Reinforcement Learning as a Rehearsal for Planning in Air Battle Management."

Publication

A paper developed under this project is under review (second round, first round decision was "major revision required") for publication in the IEEE Transactions on Games journal. It has not been published yet.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

A2C	Advantage Actor Critic
LSTM	Long short-term memory
RL	Reinforcement Learning
RLaR	Reinforcement Learning as a Rehearsal
RTS	Real Time Strategy (Games)
SIL	Self-imitation learning