Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games

Peng Peng[†], **Quan Yuan**[†], **Ying Wen**[‡], **Yaodong Yang**[‡], **Zhenkun Tang**[†], **Haitao Long**[†], **Jun Wang**[‡]
[†]Alibaba Group, [‡]University College London

Abstract

Real-world artificial intelligence (AI) applications often require multiple agents to work in a collaborative effort. Efficient learning for intra-agent communication and coordination is an indispensable step towards general AI. In this paper, we take StarCraft combat game as the test scenario, where the task is to coordinate multiple agents as a team to defeat their enemies. To maintain a scalable yet effective communication protocol, we introduce a multiagent bidirectionally-coordinated network (BiCNet ['bɪknet]) with a vectorised extension of actor-critic formulation. We show that BiCNet can handle different types of combats under diverse terrains with arbitrary numbers of AI agents for both sides. Our analysis demonstrates that without any supervisions such as human demonstrations or labelled data, BiCNet could learn various types of coordination strategies that is similar to these of experienced game players. Moreover, BiCNet is easily adaptable to the tasks with heterogeneous agents. In our experiments, we evaluate our approach against multiple baselines under different scenarios; it shows state-of-the-art performance, and possesses potential values for large-scale real-world applications.

1 Introduction

The last decade has witnessed massive progresses in the field of Artificial Intelligence (AI) [1]. With supervision from labelled data, machines have, to some extent, exceeded human-level perception on visual recognitions [2, 3] and speech recognitions [4], while fed with feedback reward, single AI units (*aka* agents) defeat humans in various games including Atari video games [5], Go game [6], and card game [7, 8].

Yet, true human intelligence embraces social and collective wisdom [9], which lays an essential foundation for reaching the grand goal of Artificial General Intelligence (AGI) [10]. As demonstrated by crowd sourcing [11], aggregating efforts collectively from the public would solve the problem which is otherwise unthinkable by a single person. Even social animals like a brood of well-organised ants could accomplish challenging tasks such as hunting, building a kingdom, and even waging a war [12], although each ant by itself is weak and limited. Interestingly, in the coming era of algorithmic economy, AI agents with a certain rudimentary level of *artificial* collective intelligence start to emerge from multiple domains. Typical examples include the trading robots gaming on the stock markets [13], ad bidding agents competing with each other over online advertising exchanges [14], and e-commerce collaborative filtering recommenders [15] predicting user interests through the wisdom of the crowd [16].

A next grand challenge of AGI is to answer how large-scale multiple AI agents could learn human-level collaborations, or competitions, from their experiences with the environment where both of their incentives and economic constraints co-exist. As the flourishes of deep reinforcement learning (DRL) [5, 17, 6], researchers start to shed light on tackling multiagent problems [18–22] with the enhanced learning capabilities.

In this paper, we leverage a real-time strategy game, *StarCraft*[1], as the use case to explore the learning of intelligent collaborative behaviours among multiple agents. Particularly, we focus on StarCraft micromanagement tasks [23], where each player controls their own units (with different functions to collaborate) to destroy the opponent's army in the combats under different terrain conditions. Such game is considered as one of the most difficult games for computers with more possible states than Go game [23]. The learning of this large-scale multiagent system faces a major challenge that the parameters space grows exponentially with the increasing number of agents involved. As such, the behaviours of the agents can become so sophisticated that any joint learner method [20] would be inefficient and unable to deal with the dynamically changeable number of agents in the game.

We formulate multiagent learning for StarCraft combat tasks as a zero-sum Stochastic Game. Agents are communicated by our proposed bidirectionally-coordinated net (BiCNet), while the learning is done using a multiagent actor-critic framework. In addition, we also introduce the concept of dynamic grouping and parameter sharing to solve the scalability issue. Our analysis shows that BiCNet can automatically learn various optimal strategies to coordinate agents, similar to what experienced human players would adopt in playing the StarCraft game, ranging from trivial *move without collision* to a basic tactic *hit and run* to sophisticated *cover attack* and *focus fire without overkill*. We have conducted our experiments by testing over a set of combat tasks with different levels of difficulties. Our method outperforms state-of-the-art methods and shows its potential usage in a wide range of multiagent tasks in the real-world applications.

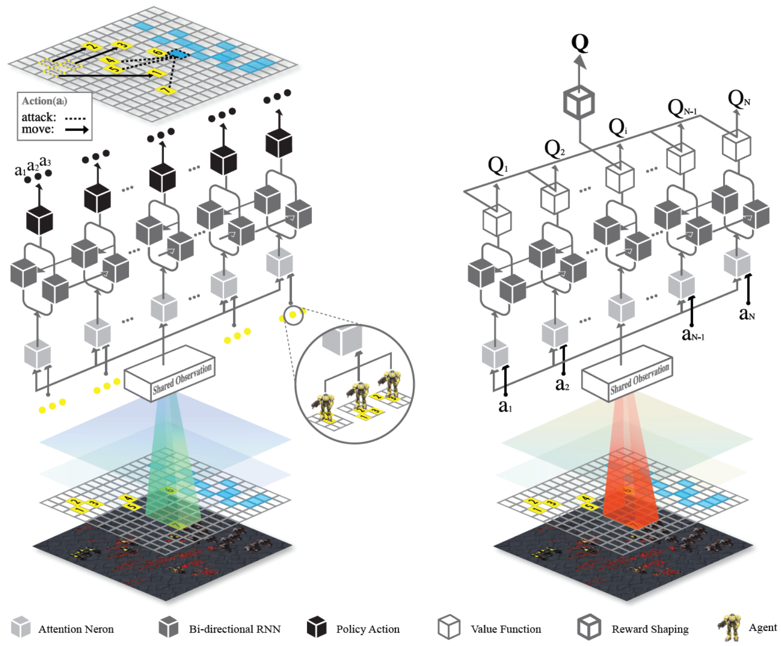
2 Related Work

The studies on interaction and collaboration in multiagent settings have a long history [24, 25, 18, 26]. Although limited to toy examples in the beginning, reinforcement learning, as a means, has long been applied to multiagent systems in order to learn optimal collaboration policies [27]. Typically, they are formalised as a stochastic game [28], and solved by minimax Q-learning [25]. As function approximators, neural networks have also been adopted and proven to be effective and flexible [18]. Nonetheless, with the increased complexity of the environment, these traditional methods no longer work well. For solving StarCraft combats, researchers resort to deep reinforcement learning (DRL) [5, 17, 6] due to the complexity of the environment and action space. For the analysis of complexity of StarCraft combat games, we refer to [29, 30]. One of the key components in using DRL is to learn a communication protocol among agents. Representative solutions include the differentiable inter-agent learning (DIAL) [19] and the CommNet [20], both of which are end-to-end trainable by back-propagation.

DIAL [19] was introduced in partially observable settings where messages passing between agents are allowed. The agent is also named as a *independent learner*. The idea of learning independent agents can also be found [31–33, 19]. In DIAL, each agent consists of a recurrent neural network that outputs individual agent's Q-value and a message to transfer for each time-step. The generated messages is then to be transferred to other agents as inputs for the next time-step. When an agent receives the messages from others, it will embed the messages together with its current observations and last action in order to take into account the overall information. However, since DIAL is designed for independent learners, it inevitably faces the challenge of not being able to tackle the non-stationary environments; in other words, the environment will keep changing for each agent. Such non-stationary problem is even more severe in real-time strategy games such as StarCraft.

By contrast, CommNet [20] is designed for *joint action learners* in fully observable settings. Unlike DIAL, CommNet proposes a single network in the multiagent setting, passing the averaged message over the agent modules between layers. However, as the communication network is fully symmetric and embedded in the original network, it lacks the ability of handle heterogeneous agent types. Also it is a single network for all agents, and therefore its scalability is unclear. In this paper, we solve these issues by creating a dedicated bi-directional communication channel using recurrent neural networks [34]. As such, heterogeneous agents can be created with a different set of parameters and output actions. The bi-directional nature means that the communication is not entirely symmetric, and the different priority among agents would help solving any possible tie between multiple optimal joint actions [35, 36].

[1]Trademark of Blizzard Entertainment™.



(a) Multiagent policy networks with grouping (b) Multiagent Q networks with reward shaping

Figure 1: Bidirectionally-Coordinated Net (BiCNet).

Multiagent systems have also been explored in more complex cases including StarCraft combat games [30]. Recent work in [37] applies the DIAL model [19] assuming agents in StarCraft are fully decentralised. The studies from [23, 30] focus on a greedy MDP approach, i.e., the action of an agent is dependent explicitly on the action of another agent that has has been generated. In this paper, the dependency of agents is, rather, modelled over hidden layers by making use of bi-directional recurrent neural networks (RNN) [34]. A significant benefit over the greedy solution is that, while keeping simple, its gradient update from all the actions is efficiently propagated through the entire networks.

3 Multiagent Bidirectionally-Coordinated Nets (BiCNet)

3.1 Preliminaries: Combat as Stochastic Games with Global Reward

The StarCraft combat, *aka* the micromanagement task, refers to the low-level, short-term control of the army members during a combat with enemy members [30]. We formulate it as a zero-sum Stochastic Game (SG) [28, 38, 39], i.e., a dynamic game in a multiple state situation played by multiple agents. A SG of N agents and M opponents (enemies in a combat) can be described by a tuple $(\mathcal{S}, \{\mathcal{A}_i\}_{i=1}^{N}, \{\mathcal{B}_i\}_{i=1}^{M}, \mathcal{T}, \{\mathcal{R}_i\}_{i=1}^{N+M}\}$. Let S denote the state space of the current game, shared among all the agents. We define the action space of the controlled agent i as $\mathcal{A}_i$. Similarly, we define the action space of the enemy j as $\mathcal{B}_j$. For simplicity, we consider the case that all agents share the same action space. That is, for each agent i $\mathcal{A}_i = \mathcal{A}$ and for each enemy j $\mathcal{B}_j = \mathcal{B}$, where $i \in [1, N]$ and $j \in [1, M]$. Denote by $\mathcal{T} : \mathcal{S} \times \mathcal{A}^N \times \mathcal{B}^M \to \mathcal{S}$ the deterministic transition function of the environment. Lastly, we denote by $\mathcal{R}_i : \mathcal{S} \times \mathcal{A}^N \times \mathcal{B}^M \to \mathbb{R}$ the reward function of agent/enemy i for $i \in [1, N + M]$.

In order to maintain a flexible framework and allow an arbitrary number of agents, we consider that all the agents, both the controlled and the enemies, share the same state space S of the current game; and within each camp, either the controlled's or the enemy's, agents are homogeneous[2] and have the same action spaces $\mathcal{A}$ and $\mathcal{B}$ respectively. Also we consider the continuous action space instead of the discrete one in this work. It significantly reduces the redundancy in modelling the large discrete

[2] We, however, demonstrate from latter section that with our framework heterogeneous agents can be also trained using different parameters and action space.

action space. In the combat, the agents in our side or the other side are fully cooperative within their own teams, but compete between teams. Thus, each combat is a zero-sum competitive game between N agents and M enemies. In this paper, we propose to use the deterministic policy $\mathbf{a}_\theta : \mathcal{S} \to \mathcal{A}^N$ of the controlled agents and the deterministic policy $\mathbf{b}_\phi : \mathcal{S} \to \mathcal{B}^M$ of the enemies.

The next is to define reward function. Reward shaping is of great importance in learning the collaborative or competing behaviours among agents [40]. In this section, as a starting point, we first consider a basic formulation with global reward (each agent in the same team shares the same reward), while leaving the definition of individual rewards and the resulting more general formulation in the next section. Specifically, we introduce a time-variant global reward based on the difference of the heath level between two consecutive time steps:

$$r(\mathbf{s},\mathbf{a},\mathbf{b}) \equiv \frac{1}{M}\sum_{j=N+1}^{M} \Delta\mathcal{R}_j^t(\mathbf{s},\mathbf{a},\mathbf{b}) - \frac{1}{N}\sum_{i=1}^{N}\Delta\mathcal{R}_i^t(\mathbf{s},\mathbf{a},\mathbf{b}), \tag{1}$$

where for simplicity, we drop the subscript t in global reward $r(\mathbf{s},\mathbf{a},\mathbf{b})$. For given time step t with state $\mathbf{s}$, the controlled agents take actions $\mathbf{a} \in \mathcal{A}^N$, the opponents take actions $\mathbf{b} \in \mathcal{B}^M$, and $\Delta\mathcal{R}_j^t(\cdot) \equiv \mathcal{R}_j^{t-1}(\mathbf{s},\mathbf{a},\mathbf{b}) - \mathcal{R}_j^t(\mathbf{s},\mathbf{a},\mathbf{b})$ represents the reduced health level for agent j. Note that Eq.(1) is presented from the aspect of controlled agents; the enemy's global reward is the exact opposite, making the sum of rewards from both camps equalling to zero. As the health level is non-decreasing over time, Eq. (1) gives a positive reward at time step t if the increase of enemies' health levels exceeds that of ours.

With the defined global reward $r(\mathbf{s},\mathbf{a},\mathbf{b})$, the controlled agents jointly take actions $\mathbf{a}$ in state $\mathbf{s}$ when the enemies take joint actions $\mathbf{b}$. The objective of the controlled agents is to learn a policy that *maximises* the expected sum of discounted rewards, i.e., $\mathbb{E}[\sum_{k=0}^{+\infty}\lambda^k r_{t+k}]$, where $0 \le \lambda < 1$ is discount factor. Conversely, the enemies' joint policy is to *minimise* the expected sum. Correspondingly, we have the following *Minimax* game:

$$Q^*_{\text{SG}}(\mathbf{s},\mathbf{a},\mathbf{b}) = r(\mathbf{s},\mathbf{a},\mathbf{b}) + \lambda \max_\theta \min_\phi Q^*_{\text{SG}}(\mathbf{s}',\mathbf{a}_\theta(\mathbf{s}'),\mathbf{b}_\phi(\mathbf{s}')), \tag{2}$$

where $\mathbf{s}' \equiv \mathbf{s}^{t+1}$ is determined by $\mathcal{T}(\mathbf{s},\mathbf{a},\mathbf{b})$. $Q^*_{\text{SG}}(\mathbf{s},\mathbf{a},\mathbf{b})$ is the optimal action-state value function, which follows the Bellman Optimal Equation.

In small-scale MARL problems, a common solution is to employ *Minimax Q-learning* [25]. However, the micromanagement combat in StarCraft is computationally expensive [30], and therefore make minimax Q-learning intractable to apply. In addition, solving the adversarial Q-function of Eq.(2) also requires an opponent/enemy modelling [35, 41]. We employ *fictitious play* [28] and deep neural nets to learn the enemies policy $\mathbf{b}_\phi$ in two situations, respectively. When the enemies are AI agents, we consider *fictitious play*, where both the controlled agents and the enemies apply empirical strategies learned from the other players so far, and then iteratively optimise the above Q-function. Alternatively, when dealing with the Game's own control of the enemies, we modelled the deterministic policy $\mathbf{b}_\phi$ through DNN in a supervised learning setting. Specifically, the policy network is trained together with sampled historical state-action pairs $(\mathbf{s},\mathbf{b})$ of the enemies:

$$\triangle\phi \propto \frac{\partial \mathbf{b}_\phi(\mathbf{s})}{\partial\phi}. \tag{3}$$

By learning the policy of enemies and fixing them, SG defined in Eq. (2) effectively turns into a simper MDP problem [41]:

$$Q^\theta(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \lambda Q^\theta(\mathbf{s}',\mathbf{a}_\theta(\mathbf{s}')), \tag{4}$$

where we drop notation $\mathbf{b}_\phi$ for brevity.

3.2 Proposed Multiagent Actor-Critic with Individual Reward

A potential drawback of the global reward in Eq. (1) and its resulting zero-sum game is that it ignores the fact that a team collaboration, typically consisting of local collaborations and reward function, would normally include certain internal structure. Moreover, in practice, each agent tends to have its own objective which drives the collaboration. To model this, we extend the formulation in the previous section by replacing Eq. (1) with each agent's local reward and including the evaluation of their attribution to other agents that have been interacting with (either completing or collaborating), i.e.,

$$r_i(\mathbf{s}, \mathbf{a}, \mathbf{b}) \equiv \frac{1}{|j|} \sum_{j=N+1 \cap \textit{top-K}(i)}^{M} \Delta \mathcal{R}_j(\mathbf{s}, \mathbf{a}, \mathbf{b}) - \frac{1}{|i'|} \sum_{i'=1 \cap \textit{top-K}(i)}^{N} \Delta \mathcal{R}_{i'}(\mathbf{s}, \mathbf{a}, \mathbf{b}) \tag{5}$$

where each agent i maintaining *top-K*(i), a top-K list of other agents that are currently being interacted. Replacing it with Eq. (1), we have N number of Bellman equations for agent i, where $i \in \{1, ..., N\}$, for the same parameter θ of the policy function:

$$Q_i^\theta(\mathbf{s}, \mathbf{a}) = r_i(\mathbf{s}, \mathbf{a}) + \lambda Q_i^\theta(\mathbf{s}', \mathbf{a}_\theta(\mathbf{s}')). \tag{6}$$

We can then write the objective as an expectation:

$$J(\theta) = \mathbb{E}_{s \sim p_1}[\sum_{i=1}^{N} r_i(s, \mathbf{a}_\theta(\mathbf{s}))] = \int_{\mathcal{S}} p_1(\mathbf{s}) \sum_{i=1}^{N} Q_i^\theta(\mathbf{s}, \mathbf{a})|_{\mathbf{a}=\mathbf{a}_\theta(\mathbf{s})} \mathrm{d}\mathbf{s}. \tag{7}$$

In continuous action spaces, the classical model-free *policy iteration* methods become intractable, as the greedy policy search requires a global maximisation at every step. Here we develop a vectorised version of deterministic policy gradient (DPG) on the basis of [17, 42, 43] for searching the policy in the direction of the gradient of the sum of Q_i instead of maximising it. Taking the derivative of Eq. (7) gives:

$$\nabla_\theta J(\theta) = \frac{\partial}{\partial \theta} \int_{\mathcal{S}} p_1(\mathbf{s}) \sum_{i=1}^{N} Q_i^{\mathbf{a}_\theta}(\mathbf{s}, \mathbf{a}) \mathrm{d}\mathbf{s} = \mathbb{E}_{\mathbf{s} \sim \rho^{\mathcal{T}}_{\mathbf{a}_\theta}(\mathbf{s})} \left[\sum_{j}^{N} \left(\sum_{i=1}^{N} \frac{\partial Q_i^{\mathbf{a}_\theta}(\mathbf{s}, \mathbf{a})|_{\mathbf{a}=\mathbf{a}_\theta(\mathbf{s})}}{\partial \mathbf{a}_j} \right) \frac{\partial \mathbf{a}_{j,\theta}(\mathbf{s})}{\partial \theta} \right], \tag{8}$$

where the gradients pass to both individual Q_i function and the policy function. They are aggregated from all the agents and their actions. In other words, the gradients from all agents rewards are first propagated to influence each of agent's actions, and the resulting gradients are further propagated back to updating the parameters.

To ensure adequate exploration, we considered the off-policy deterministic actor-critic algorithms [17, 42, 43]. Particularly, we employ a *critic* to estimate the action-value function where off-policy explorations can also be conducted. We thus have the following gradient for learning the parameters of *critic* $Q^\xi(\mathbf{s}, \mathbf{a})$ aggregated from multiple agents:

$$\nabla_\xi L(\xi) = \mathbb{E}_{\mathbf{s} \sim \rho^{\mathcal{T}}_{\mathbf{a}_\theta}(\mathbf{s})} \left[\sum_{i=1}^{N} \left(r_i(\mathbf{s}, \mathbf{a}_\theta(\mathbf{s})) + \lambda Q^\xi(\mathbf{s}', \mathbf{a}_\theta(\mathbf{s}')) - Q^\xi(\mathbf{s}, \mathbf{a}_\theta(\mathbf{s})) \right) \frac{\partial Q_i^\xi(\mathbf{s}, \mathbf{a}_\theta(\mathbf{s}))}{\partial \xi} \right]. \tag{9}$$

With Eqs. (8) and (9), we are ready to use Stochastic Gradient Descent (SGD) to compute the updates for both the actor and the critic networks. The pseudocode of our proposed algorithm can be founded in the supplementary.

We next introduce our design for the two networks. For each agent, we share their parameters so that the number of parameters is independent of the number of agents. The resulting more compact model would be able to learn over various situations experienced from multiple agents, and thus speed up the learning process. Additionally, as later demonstrated in our experiment, such shared parameter space would allow us to train using only a smaller number of of agents (typically three), while freely scaling up to any larger number of agents during the test.

To make effective communication between agents, we introduce dedicated bi-directional connections in the internal layers by employing bi-directional recurrent neural networks (RNN) [34]. As we discussed, it is different with the greedy approach from [23, 30] in that the dependency of our agents are built upon the internal layers, rather than directly from the actions. While simple, our approach allow full dependency among agents because the gradients from all the actions in Eq. (9) are efficiently propagated through the entire networks. Yet, unlike CommNet [20], our communication is not fully symmetric, and we maintain certain social conventions and roles by fixing the order of the agents that join the RNN. This would help solving any possible tie between multiple optimal joint actions [35, 36].

The structure of our bidirectionally-coordinated net (BiCNet) is illustrated in Fig. 1. It consists of the policy networks (actor) and the Q-networks (critic), both of which are based on bi-directional RNN. The policy networks, taking a shared observation together with a local view, return actions for individual agents. Therefore, individual agents are able to maintain their own internal states, while being able to share the information with other collaborators. The bi-directional recurrent mechanism acts not only as a communication means between agents but also as a local memory state. We further intensify the actor with the concept of grouping [44], which plays an important role in influencing social behaviours. As an input we allow a small number of agents to build a local correlation before fed into the recurrent layers. As such, the model scales much better. In the experiments, we found that the group-based actor can help learning the group activities such as focusing fire. The Q-value networks, taking the state and the actions from the policy networks as inputs, return the estimated local Q-value for each individual agent. The local Q-value is then combined to provide the estimation of the global reward.

4 Experiments

4.1 Experimental Setup

To understand the properties of our proposed BiCNet and its performance, we conducted experiments over different settings of the StarCraft combats. Following similar experiment set-up as [20], BiCNet controls a group of agents trying to defeat the enemy units controlled by the built-in AI. The level of combat difficulties can be adjusted by varying the unit types and the number of units in both sides. We measured the winning rates, and compared it with the state-of-the-art approaches. The comparative baselines consist of both the rule-based approaches and deep reinforcement learning approaches. Our setting is summarised as follows where BiCNet controls the former units and the built-in AI controls the latter, including easy combats {*3 Marines vs. 1 Super Zergling*, and *3 Wraiths vs. 3 Mutalisks*}, difficult combats {*5 Marines vs. 5 Marines*, *15 Marines vs. 16 Marines*, *20 Marines vs. 30 Zerglings*, *10 Marines vs. 13 Zerglings*, and *15 Wraiths vs. 17 Wraiths*.}, heterogeneous combats { *2 Dropships and 2 Tanks vs. 1 Ultralisk* }.

Despite the fact that rule-based approaches are usually trivial, they allow us to have a reference point that we could make sense of. Here we adopted three rule-based baselines: **StarCraft built-in AI**, **Attack the Weakest**, **Attack the Closest**.

For the deep reinforcement learning approaches, we considered the following the baselines:

Independent controller (IND): We trained the model for single agent and control each agent individually in the combats. Note that there is no information sharing among different agents even though such method is easily adaptable to all kinds of multiagent combats.

Fully-connected (FC): We trained the model for all agents in a multiagent setting and control them collectively in the combats. The communication between agents are fully-connected. We need to re-train a different model when the number of units at either side changes.

CommNet: CommNet [20] is a multiagent network designed to learning to communicate among multiple agents. To make a fair comparison, we implement bot the CommNet and the BiCNet on the same (state, action) spaces and follow the same training processes.

GreedyMDP with Episodic Zero-Order Optimisation(GMEZO): GMEZO [30] was proposed particularly to solve StarCraft micromanagement tasks. Apart from traditional DRL approaches, two novel ideas are introduced: conducting collaborations through a greedy update over MDP agents, as well as adding episodic noises in the parameter space for explorations. To focus on the comparison with these two ideas, we replaced our bi-directional formulation with the greedy MDP approach, and employed episodic zero-order optimisation with noise over the parameter space in the last layer of Q networks in our BiCNet.

It is also worth mentioning that we define the action space and the state space differently from the previous works [20]. Specifically, the action space of each individual agent is represented as a 3-dimensional real vector. The first dimension ranges from -1 to 1. When its value is greater than or equal to 0, the agent attacks; otherwise, the agent moves. The second dimension and the third dimension correspond to the degree and the distance, collectively indicating the destination that the agent should move or attack from its current location. The state space of each individual agent is

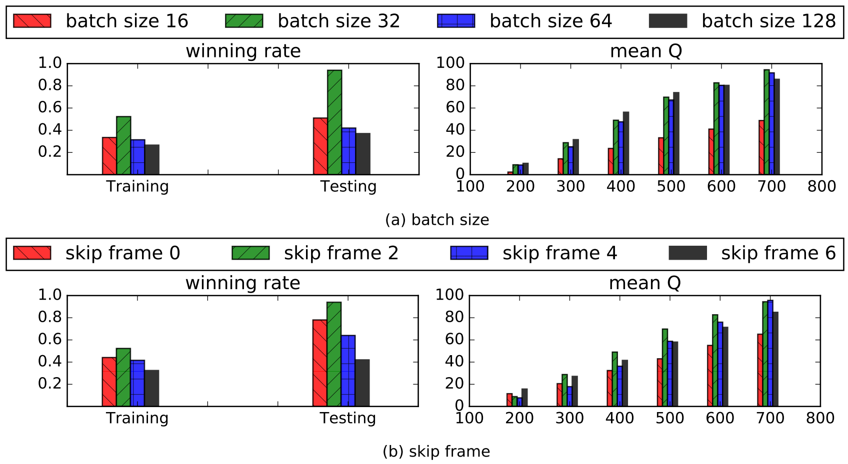


Figure 2: The impact of **batch_size** and **skip_frames** in combat *2 Marines vs. 1 Super Zergling*.

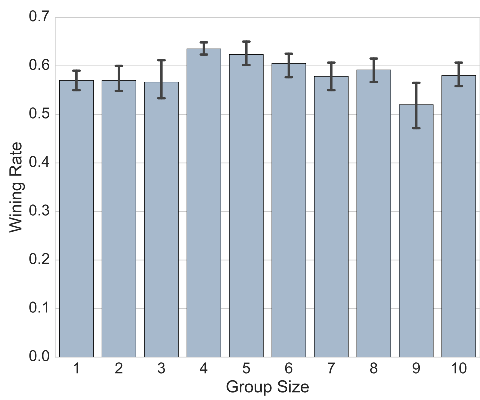


Figure 3: Winning rate vs. group size in combat "10 *Marines* vs. 13 *Zerglings*.

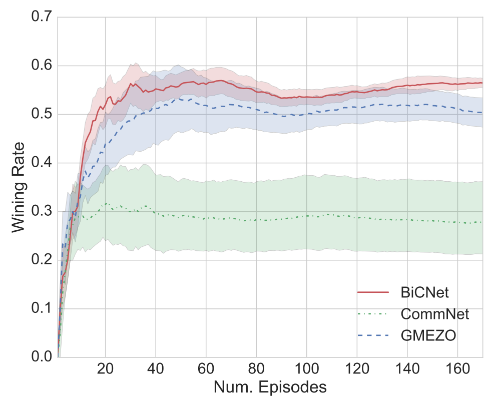


Figure 4: Learning Curves in Combat "10 *Marines* vs. 13 *Zerglings*"

represented as a tensor of size $72 \times 72 \times 16$ extracted from the map of size 72×72. Each channel in the state space describes either the hit points, the damage or the safety attacking range for all the agents on the map.

4.2 Parameter Tuning

In our training, *Nadam* [45] is set as the optimiser with learning rate equal to 0.002 and the other arguments set by default values. We set the maximum steps of each episode as 800.

We first study the impact of the batch size and skip frame (how many frames we should skip for controlling the agents actions). Figure 2 gives the results of the parameter tuning in the "2 Marines vs. 1 Super Zergling" combat. The two metrics, the winning rate and the Q value, are given. We fine-tune the *batch_size* and the *skip_frame* by selecting the best BiCNet model which are trained on 800 episodes (more than 700k steps) and then tested on 100 independent games. As suggested in the figures, when *batch_size* is 32 and *skip_frame* is 2, the model has the highest winning rates. Also, the model with *batch_size* 32 achieves the highest mean Q-value after 600k training steps, while the model with skip frame 2 has the highest mean Q-value between 300k and 600k training steps. We fix both parameters with the learned optimal values in the remaining of our test.

Next, we analyse the impact of the dynamic group size on the winning rate. Empirical results in Figure 3 show that setting 4-5 as group size would help achieve best performance. This is consistent with the discussion about the focus fire strategy in Section A.2, where we discover that letting 4-6

Table 1: Performance comparison. M: *Marine*, Z: *Zergling*, W: *Wraith*.

Combat	Rule Based			RL Based				
	Built-in	Weakest	Closest	IND	FC	GMEZO	CommNet	BiCNet
20 M vs. 30 Z	**1.00**	.000	.870	.940	.001	.882	**1.00**	**1.00**
5 M vs. 5 M	.720	.900	.700	.311	.076	.909	**.950**	.920
15 M vs. 16 M	.610	.000	.670	.590	.438	.627	.682	**.712**
10 M vs. 13 Z	.550	.230	.410	.522	.430	.570	.440	**.640**
15 W vs. 17 W	.440	.000	.300	.310	.460	.420	.470	**.531**

agents work together as a group can efficiently control individual agents while maximising the damage output.

4.3 Performance Comparison

Table 1 compares our proposed BiCNet model against the baselines under multiple combat scenarios. In each scenario, BiCNet is trained over 100k steps, and we measure the performance as the average winning rate on 100 test games. The winning rate of the built-in AI is also provided as an indicator of the level of difficulty of the combats.

As illustrated in Table 1, in $4/5$ of the scenarios, BiCNet outperforms the other baseline models. In particular, when the number of agents goes beyond 10, where cohesive collaborations are required, the margin of performance between BiCNet and the second best starts to increase. Apart from the winning rate, in Fig. 4, we also compare the convergence speed by plotting the winning rate against the number of training episodes. It shows that the proposed BiCNet model has a much quicker convergence than the rest.

In combat "5 M vs. 5 M", where the key factor to win is to "focus fire" on the weak, the IND and the FC models have relatively poorer performance. We believe it is because both of the models do not come with an explicit collaboration mechanism between agents in the training stage; Coordinating the attacks towards one single enemy is even challenging. On the contrary, GMEZO, CommNet, and BiCNet, which are designed for the multiagent collaboration, can grasp and master this simple strategy, thus enjoying better performances. As BiCNet has built-in design for dynamic grouping, small number of agents (such as the case "5 M vs. 5 M") does not suffice to show the advantages of BiCNet on large-scale collaborations. Furthermore, it is worth mentioning that despite the second best performance on "5 Marines vs. 5 Marines", our BiCNet only needs 10 combats before learning the idea of "focus fire", and achieves over 85% win rate, whereas CommNet needs more than 50 episodes to grasp the skill of "focus fire" with a much lower winning rate.

4.4 Visualization

For further investigating BiCNet, we visualize the results based on "3 Marines vs. 1 Super Zergling" when the coordinated cover attack has been learned. We collect the values in the last hidden layer of the well-trained critic network over **10k** steps and embed them in 2-dimensional space using t-SNE algorithm [46]. We observe that the steps with high Q-values are aggregated in the same area in the embedding space. For example, the left-most figure in Fig.5 shows that the agents attack the enemy in long distance where the enemy cannot attack the agents immediately. It corresponds to a point with high Q-values in the 2-dimensional space shown in the middle figure. On the other hand, in the right-most figure in Fig. 5 displays an instance that the agents are attacked by the enemy in a close range. We observe such a case corresponds to a point with low Q-value in the 2-dimensional space. More experiments with various combat scenarios are included in the supplementary, in which readers can refer to.

5 Conclusions

In this paper, we have introduced a new deep multiagent reinforcement learning framework, making use of bi-directional neural networks. The collaboration is learned by constructing a vectorised actor-critic framework, where each dimension corresponds to an agent. The coordination is done by bi-directional communications in the internal layers. Through end-to-end learning, our BiCNet

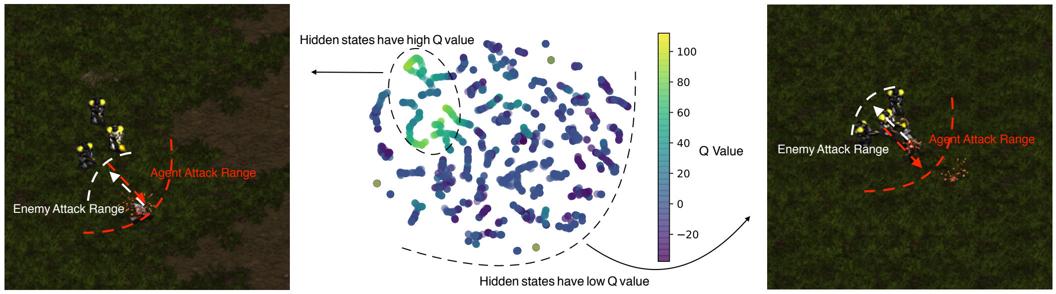


Figure 5: Visualisation for 3 Marines vs. 1 Super Zergling combat. **Left**: a case with high Q-value; **Middle**: Visualisation of hidden layer outputs using t-SNE, coloured in terms of their Q-values; **Right**: a case with low Q value.

would be able to successfully learn several effective coordination strategies. Our experiments have demonstrated its ability to collaborate and master diverse combats in StarCraft combat games.

In our experiments, we found that there was a strong correlation between the specified rewards and the learned policies. We plan to further investigate the relationships and study how the policies are communicated over the networks among agents, and whether there is a specific language that may have emerged [21, 22]. In addition, it is of interest to explore Nash equilibrium when both sides are played by deep multiagent models.

References

[1] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[2] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[4] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[6] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[7] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

[8] Noam Brown and Tuomas Sandholm. Safe and nested endgame solving for imperfect-information games. *AAAI/IAAI*, 2017.

[9] Marquis de Condorcet. Essay on the application of analysis to the probability of majority decisions. *Paris: Imprimerie Royale*, 1785.

[10] Ben Goertzel and Cassio Pennachin. *Artificial general intelligence*, volume 2. Springer, 2007.

[11] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.

[12] Andrew FG Bourke and Nigel R Franks. *Social evolution in ants*. Princeton University Press, 1995.

[13] Guido Deboeck. *Trading on the edge: neural, genetic, and fuzzy systems for chaotic financial markets*, volume 39. John Wiley & Sons, 1994.

[14] Jun Wang, Weinan Zhang, and Shuai Yuan. Display advertising with real-time bidding (RTB) and behavioural targeting. *Foundations and Trends® in Information Retrieval, Now Publishers*, 2017.

[15] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[16] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

[17] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[18] Jurgen Schmidhuber. A general method for multi-agent reinforcement learning in unrestricted environments. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 84–87, 1996.

[19] Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[20] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[21] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

[22] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

[23] Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*, 2016.

[24] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[25] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, pages 157–163, 1994.

[26] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998:746–752, 1998.

[27] Norihiko Ono and Kenji Fukumoto. Multi-agent reinforcement learning: A modular approach. In *Second International Conference on Multiagent Systems*, pages 252–258, 1996.

[28] Guillermo Owen. *Game theory*. Academic Press, 1995.

[29] Santiago Ontanón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. A survey of real-time strategy game ai research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in games*, 5(4):293–311, 2013.

[30] Nicolas Usunier, Gabriel Synnaeve, Zeming Lin, and Soumith Chintala. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint arXiv:1609.02993*, 2016.

[31] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

[32] Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. *AAAI/IAAI*, 2002:326–331, 2002.

[33] Martin Lauer and Martin Riedmiller. Reinforcement learning for stochastic cooperative multi-agent systems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1516–1517. IEEE Computer Society, 2004.

[34] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[35] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews*, 38(2):156, 2008.

[36] Matthijs TJ Spaan, Nikos Vlassis, Frans CA Groen, et al. High level coordination of agents based on multiagent markov decision processes with roles. In *IROS*, volume 2, pages 66–73, 2002.

[37] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Philip Torr, Pushmeet Kohli, Shimon Whiteson, et al. Stabilising experience replay for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1702.08887*, 2017.

[38] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.

[39] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[40] Adrian K Agogino and Kagan Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, 17(2):320–338, 2008.

[41] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1804–1813, 2016.

[42] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[43] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[44] Richard D Alexander. The evolution of social behavior. *Annual review of ecology and systematics*, 5(1):325–383, 1974.

[45] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147, 2013.

[46] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

A Appendix

A.1 Pseudocode

Algorithm 1 BiCNet algorithm

Initialise actor network $\mathbf{a}_\theta(\mathbf{s})$ and critic network $Q^\xi(\mathbf{s}, \mathbf{a})$ with parameters ξ and θ
Initialise target network $\mathbf{a}_{\theta'}(\mathbf{s})$ and critic network $Q^{\xi'}(\mathbf{s}, \mathbf{a})$ with $\xi' \leftarrow \xi$ and $\theta' \leftarrow \theta$
Initialise replay buffer R
for episodes=1, M **do**
  initialise a random process $\mathcal{N}$ for action exploration
  receive initial observation state s_1
  for t=1, T **do**
    select and execute action $a_i^t = \mathbf{a}_\theta(s_t) + \mathcal{N}_t$ for each agent$i$
    receive reward r_i^t and observe new state s_i^{t+1}
    store transition $[s_i^t, a_i^t, r_i^t, s_i^{t+1}]_{i=1}^N$ in R
    sample a random minibatch of M transitions from R
    compute target value for each agent in each transition $(\hat{Q}_1^t, \hat{Q}_2^t, ..., \hat{Q}_N^t)$ using the Bi-RNN:

$$\hat{Q}_i^t = r_i^t + \gamma Q^{\xi'}(s_i^{t+1}, \mathbf{a}_{\theta'}(s_i^{t+1}))$$

    compute critic update according to Eq.(9) via BPTT:

$$\Delta\xi = \frac{1}{M}\sum_{m=1}^{M} \nabla_\xi L(\xi)$$

    compute actor update according to Eq.(8) via BPTT:

$$\Delta\theta = \frac{1}{M}\sum_{m=1}^{M} \nabla_\theta J(\theta)$$

    update the target network:

$$\xi' \leftarrow \gamma\xi + (1-\gamma)\xi', \qquad \theta' \leftarrow \gamma\theta + (1-\gamma)\theta'$$

  end for
end for

A.2 Analysis of Learned Coordination Strategies

With adequate trainings from scratch, BiCNet would be able to discover several effective collaboration strategies. In this section, we conduct a qualitative analysis on the collaboration policies that BiCNet has learned. For the detailed experiment configurations (as well as the parameter tuning and the performance comparisons), we refer to Section 4.

Coordinated moves without collision. We observe that, in the initial stages of learning (as illustrated in Fig. 6 (a) and (b)), the agents move in a rather uncoordinated way, particularly, when two agents are close to each other, i.e., one agent may unintentionally block the other's path. With the increasing rounds of training (typically

Table 2: Winning rate against difficulty settings by hit points (HP) and damage. Training steps: 100k/200k/300k.

Difficulty	Damage=4			Damage=3		
	100K	200k	300k	100K	200k	300k
HP=210	0.750/1.	1./1.	1./1.	0.993/1.	1./1.	1./1.
HP=270	.140/.490	.342/.470	.670/1.	.951/1.	.873/.750	.733/.650

over 40k steps in around 50 episodes in the "3 Marines vs. 1 Super Zergling" combat setting), the number of collisions reduces dramatically. Finally, when the training bocame stable, the coordinated moves emerge, as illustrated in the right two scenes in Fig. 6. Such coordinated moves are also demonstrated in Fig. 10 (a) and (b).

Hit and Run tactics. For human players, a common tactic of controlling agents in StarCraft combat is *Hit and Run*, i.e., move agents away if under attack, and fight back when feel safe again. We find that BiCNet can rapidly grasp the tactic of *Hit and Run*, either in the case of single agent or multiple agents settings. This is illustrated in Fig. 7. Despite the simplicity of *Hit and Run*, it serves as the basis for more advanced and sophisticated collaboration tactics to be explained next.

Coordinated cover attack. *Cover attack* is a high-level collaborative strategy that is often used on the real battlefield. The essence of cover attack is to let one agent draw fire or attention from the enemies. At the meantime, other agents can take advantage of the time or distance gap to cause more harms. The difficulty of conducting cover attack lies in how to turn the moves of multiple agents into a coordinated sequential *hit and run* moves. As shown in Figs. 8 and 9, BiCNet can master it well.

In Fig. 8 time step 1, BiCNet controls the bottom two *Dragoons* to run away from the enemy *Ultralisk*, while the one in the upper-right corner immediately starts to attack the enemy *Ultralisk* (e.g., cover them up). As a response, the enemy starts to attack the top one in time step 2. The bottom two *Dragoons* fight back and form another cover-up, consequently. By continuously looping this over, the team of *Dragoons* guarantees consecutive attack outputs to the enemy while minimising the team-level damages (because the enemy wastes time in targeting different *Dragoons*) until the enemy is killed.

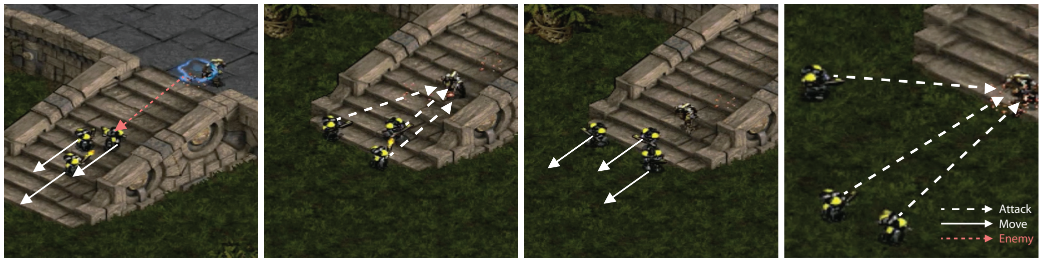
Interestingly, we also find that the *coordinated cover attack* strategy appears only for combats with a certain level of difficulty. In StarCraft combat, the difficulty of a combat can be adjusted by changing the number of enemies and the *hit points* and the *damage* of each enemy. The *hit points* of a unit is referred to the amount of damage this unit may take before it is destroyed; the *damage* of a unit is the loss of hit points made by this unit when attacking others. Table 2 gives the winning rates under the different configurations of the hit points and damage.

In combat "3 Marines vs. 1 Super Zergling" (shown in Figure 9), we fix the number of enemies (i.e., 1) and adjust the difficulty by changing the hit points and the damage of *Zergling*. In this case, BiCNet only masters the policy of *cover attack* when the hit points of the *Zergling* are higher than 210 and the damage is set to 4 — the original hit points and damage of a *Zergling* are 35 and 5, respectively.

Focus fire without overkill. As the number of agents increases, how to efficiently allocate the attacking resources becomes important. Neither scattering over all enemies nor focusing on one enemy (wasting attacking

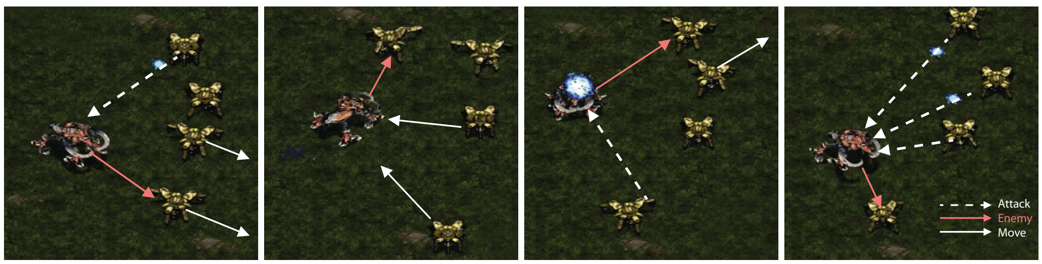
(a) Early stage of training (b) Early stage of training (c) Well-trained (d) Well-trained

Figure 6: Coordinated moves without collision in combat *3 Marines (ours) vs. 1 Super Zergling (enemy)*. The first two (a) and (b) illustrate that the collision happens when the agents are close by during the early stage of the training; the last two (c) and (d) illustrate coordinated moves over the well-trained agents.



(a) time step 1: run when attacked (b) time step 2: fight back when safe (c) time step 3: run again (d) time step 4: fight back again

Figure 7: *Hit and Run* tactics in combat *3 Marines (ours) vs. 1 Zealot (enemy)*.

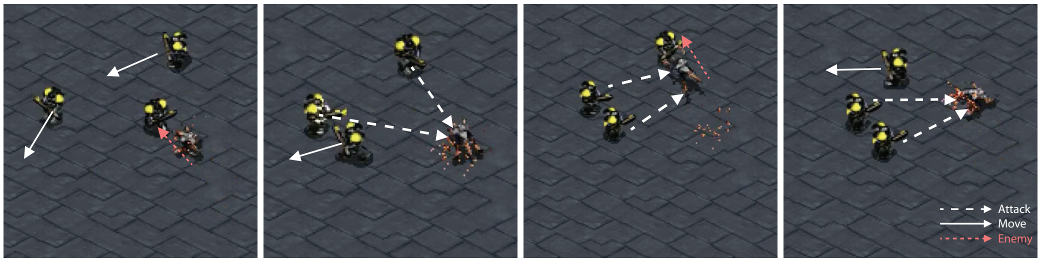


(a) time step 1 (b) time step 2 (c) time step 3 (d) time step 4

Figure 8: Coordinated cover attack in combat *4 Dragoons (ours) vs. 2 Ultralisks (enemy)*.

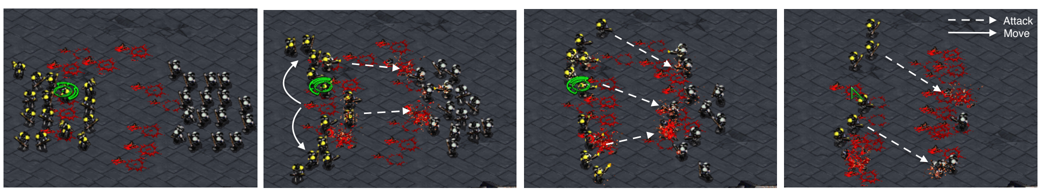
fires is also called overkill) are desired. The grouping design in the policy network serves as the key factor for BiCNet to learn "focus fire without overkill". In our experiments, we dynamically group the agents based on agents' geometric locations. Based on the grouping inputs, BiCNet manages to capture the intra-group behaviour and the inter-group behaviour among the agents. For the agents from the same group, their behaviours are generally consistent and are expected to concentrate their fires on one or two enemies. For the agents from different groups, they are expected to focus fire on different enemies. In Fig. 10, we set up group size as 6 in the "15 Marines vs. 16 Marines" battle; our units were roughly assigned to 3 groups. We found that the agents learn to focus their fires on attacking 2-3 enemies, and different groups of agents are able to learn to move to the sides to spread the fire. Even with the decreasing of our unit number, each group can be dynamically resigned to make sure that the 3-5 units focus on attacking one same enemy.

Collaborations between heterogeneous agents. In StarCraft, there are tens of types of agent units, each with unique functionalities, action space, strength, and weakness. For a combat with different types of units involved, we still expect the collaborations can be done based on their attributes. In fact, heterogeneous collaborations can be easily implemented in our framework by limiting the parameter sharing only to the same types of the units. In this paper, we study a simple case where two *Dropships* and two *tanks* collaborate to fight against an *Ultralisk*. A *Dropship* does not have the function of attack, but it can carry maximally two ground units in the air. As shown in Fig. 11, when the *Ultralisk* is attacking one of the *tanks*, the *Dropship* escorts the *tank* to escape from the attack. At the same time, the other *Dropship* unloads his *tank* to the ground so as to attack the *Ultralisk*. At each side, the collaboration between the *Dropship* and the *tank* keeps iterating until the *Ultralisk* has been destroyed.



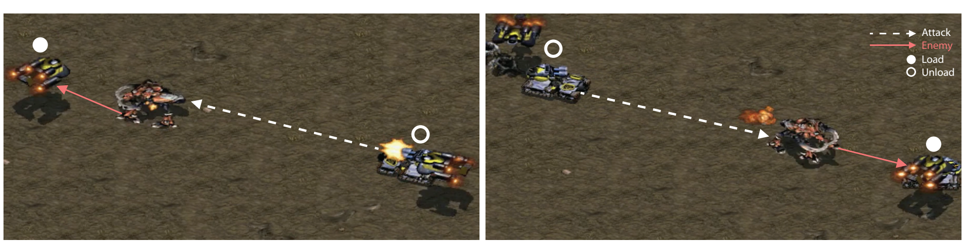
(a) time step 1 (b) time step 2 (c) time step 3 (d) time step 4

Figure 9: Coordinated cover attack in combat *3 Marines (ours) vs. 1 Zergling (enemy)*.



(a) time step 1 (b) time step 2 (c) time step 3 (d) time step 4

Figure 10: "focus fire" in combat *15 Marines (ours) vs. 16 Marines (enemy)*.



(a) time step 1 (b) time step 2

Figure 11: Coordinated heterogeneous agents in combat 2 *Dropships* and 2 *tanks* vs. 1 *Ultralisk*.