Swarm Q-Learning With Knowledge Sharing Within Environments for Formation Control

Tung Nguyen, Hung Nguyen, Essam Debie, Kathryn Kasmarik, Matthew Garratt, and Hussein Abbass
School of Engineering and IT
UNSW-Canberra
Canberra, Australia
{Duy.Nguyen,Hung.Nguyen}@student.adfa.edu.au,{e.debie,k.kasmarik,m.garratt,h.abbass}@adfa.edu.au

***Abstract*— A formation is a geometric shape that a group of agents spatially organizes themselves into and maintains over time. Swarm Q-Learning (SQL) is a tabular multi-agent reinforcement learning algorithm designed to solve formation control problems. We modify SQL by allowing agents to exchange knowledge they have learnt within the same environment and introduce the Swarm Q-Learning with knowledge Sharing wIthin an Environment (SQL-SIE). The algorithm is tested on a task where a swarm of robots, initially scattered in one side of the environment, needs to navigate through obstacles until they reach their initial positions in the formation within a region of interest. Experimental results show that the proposed SQL-SIE is more efficient than SQL as measured by the time taken by the swarm to complete this part of the mission. Moreover, SQL-SIE scales better than SQL as the number of agents increases.**

***Index Terms*— Multi-agent Formation Control, Reinforcement Learning, Task Decomposition, Knowledge Sharing**

I. Introduction

In Nature, self-organized behaviors have been a fascinating topic of research in artificial life, control theory, and computational intelligence. The behaviors of social insects and birds are characterized with perceived patterns (formations) that have sparked significant amount of research in multi-agent systems, robotics and swarm intelligence on formation control [1]. Reasons for the existence of these patterns have been attributed to their contribution to the robustness of the overall system, a decrease in groups' operating cost, ability to create and switch between flexible geometries, and allowing the group to self-reconfigure [2]–[5]. A practical example of significance in today's world is the need for formation control of small satellites to decrease energy consumption and augment the hardware capabilities of individual units [6]. The applications extend to many real-life situations, with a significant interest in the area of cooperative control of a group of unmanned aerial and ground autonomous robots. The geometric constraints on the formation are an important feature in a wide range of applications including reconnaissance in military missions [7] and search and rescue missions [8] in hazardous situations.

The problem has been investigated in the literature. Underneath the surface of that literature lies the classic multi-agent systems control problem which is still to today an active area of research [9], [10]. A second level of complexity in multi-agent control is formation consent, where agents need to move while maintaining geometric constraints. While attempts to hardwire controllers to solve this problem have been made, the prospect that learning algorithms bring to solve the problem in complex dynamic environments attracted attention of machine learning researchers. This problem can be decomposed into two phases. The first phase is for each agent to move to the right position, thus, ensuring that the geometric constraints are satisfied. This phase has been investigated during a navigation task, where agents need to get from their starting positions to their goal positions while simultaneously navigating through an environment and avoiding the obstacles they encounter on the way to their target positions [11].

The second phase is where the agents need to move without breaking up the integrity of the initial formation. If a formation breaks down, the process starts again by applying the first phase to get back into order then the second phase to maintain order. An example of work on the second phase assumes the agents start from a valid formation and need to continue moving while maintaining that formation [12]. This paper aims at addressing the first phase of a formation control problem.

In the literature, two research themes exist on formation control. One is behavioral based, while the other is control-based. An example of the former is the work of Xu et al. [13], where the authors propose a set of fixed rules for all agents to use to decide on their movement patterns to maintain a formation until they reach a target area. An example of the latter is the work of Guillet et al. [14] and Sen et al. [15]. Most of the work in this second theme rely on a predefined formation template; thus, the formation control architecture is predefined. All agents need to adhere to the formation defined by this template during operation. The previous research lacks flexibility and gets confronted with challenges when the operating environment is uncertain, large, and require adaptive formations.

Recently, there is significant amount of research focusing on solving multi-agent problems using machine learning. Machine learning approaches offer the flexibility to adapt to different situations by learning how to achieve a task that was not known at the design stage. The multi-agent reinforcement learning framework [16]–[19] uses reinforcement learning (RL) [20] to learn an optimal policy through interactions with the environment. The RL approach sits in-between the supervised learning approach, where the experience/observation is

978-1-5090-6014-6/18/$31.00 ©2018 IEEE

given by an expert, and the unsupervised learning approach, where no feedback is provided but the learning process is guided with a pre-designed cost function.

In RL, each agent receives an instantaneous (after each action) or a delayed (after a group of actions) feedback from the environment. The feedback is quantified and take the form of a scalar representing the magnitude of the reward given to the agent for its behaviour. The overall objective of the learning process is for the agent to identify which strategy it should use to maximize its accumulated reward over time. A strategy is a state-action map that allows the agent to know what actions to perform in each state. The Q-learning algorithm [21] is a model-free algorithm that attempts to learn the usefulness/values of each state-action pair.

Iima and Kuroe [22] discussed two algorithms for Q-learning in a swarm setting, primarily focusing on phase 1 of the formation control problem. The first one is a Q-learning algorithm that each agent uses independently but the reward function takes into account the overall team behaviour. We will call this version Swarm Q-Learning (SQL). The second version allows these agents to operate in multiple worlds/environments and exchange information between these worlds using what they called superior Q-values. We will call this version Swarm Q-Learning with knowledge Sharing Across Environments (SQL-SAE).

When prototyped in the work conducted for the current manuscript, we discovered that both approaches do not scale when the state space increases. Significant amount of computational resources are needed for the swarm to discover appropriate policies. Moreover, training on multiple environments increase the computational cost as well. To overcome these limitations, we propose to share knowledge within an environment to speed up the learning process. We will refer to our method as Swarm Q-Learning with knowledge Sharing wIthin Environments (SQL-SIE).

Section II gives the problem definition, Section III summarizes SQL, followed by our proposed algorithm, SQL-SIE in Section IV. Experimental results and conclusions are discussed in Sections V and VI, respectively.

II. Problem Definition

The context of this paper is phase 1 of the formation control problem as schematically depicted in Figure 1. The target positions are organized geometrically to match the desired formation.

Given an $M_1 \times M_2$ two-dimensional environment, with M_1 representing the width and M_2 representing the length of the environment, K agents, and equally K targets as we assume there is one target that each agent needs to reach, an initial base area where all agents are initially located and a target area, we call region of interest (ROI), where all targets are located, B obstacles spread in the area between the base area and the area of targets, the problem is to find the best policy for K agents to navigate from their initial position to the target positions located within ROI, while avoiding the obstacles. For simplicity, and without loss of any generality, we assume that the base area is situated at the bottom left of the environment and the target area is located at the top right end of the environment.

Each agent $I_k \in \{I_1, I_2, ..., I_K\}$ starts from its original position denoted by $O_k(x, y)$ and is required to navigate towards its target position in the set of locations $\{F_1, F_2, ..., F_K\}$. Let x_k and y_k be the current coordinates of agent I_k. Let L be the number of possible actions. All agents act synchronously and stop when they reach their target position or when the maximum amount of time allocated for the simulation has been reached. An agent that chooses to move to a cell that is occupied with an obstacle remain in its current position unchanged.

Let J be the number of combinations of possible individual agents' policies, π_k^j be the policy of agent I_k in combination $1 \leq j \leq J$ to reach a target, and $N(\pi_k^j)$ be the number of transitions in the policy π_k^j, the optimal policy for multi-agents $\pi_{multi}^{j^*}$ is a set of agents' policies defined as $\pi_{multi}^{j^*} = \{\pi_1^{j^*}, \pi_2^{j^*}, ..., \pi_K^{j^*}\}$. The aim of this problem is to find the optimal combination of policies to satisfy Equation 1.

$$j^* = argmin_j\{max[N(\pi_1^j), N(\pi_2^j), ..., N(\pi_K^j)]\}, \forall j \quad (1)$$

In this paper, we consider the number of steps as a measure of time and assume four actions ($L = 4$) to be executed: moving up $(x_k, y_k + 1)$, moving down $(x_k, y_k - 1)$, moving left $(x_k - 1, y_k)$, and moving right $(x_k + 1, y_k)$.

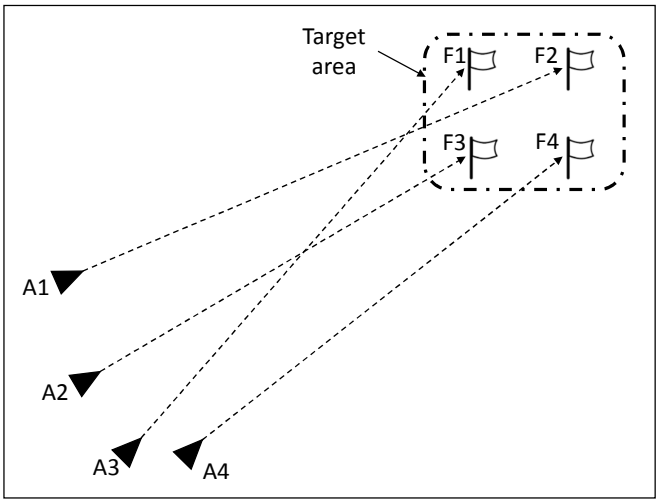

Fig. 1. Formation Initiation problem for multi-agents.

III. Classic Q-Learning Method for Multi-Agent Problem

In this section, we briefly review SQL algorithm. In a classic static environment, Q-learning algorithm has a convergence guarantee which is mentioned in [21]. In RL, an action selection method has to consider the trade-off between exploration and exploitation. In this paper, epsilon-greedy (ϵ-greedy) is used as the action-selection method. The ϵ starts from 1 to

facilitate exploration and then reduces gradually afterwards for more greedily exploitation and acceleration of the learning algorithm.

The success of a task in a multi-agent setting is evaluated by the performance of all agents. This is achieved by giving team-level performance more weight using the SQL algorithm introduced in [22]. Algorithm 1 describes the procedure of SQL.

The process above is for each single agent attempting to find the individual optimal policy. However, to trade-off the best policy for an individual and the benefit for the entire team, the reward scheme is modified as follows: at the end of each episode, the states of the agents are considered for a bonus reward r_{bonus}. If all agents occupy all target coordinates respectively, each of them receives a positive reward $r_{bonus} = r^{+}$. If there are remaining unoccupied targets, the agent which has occupied a target receives a minor negative reward $r_{bonus} = r^{-}$; otherwise the agent which has not occupied any target position receives a major negative reward $r_{bonus} = r^{-}$. By providing a multi-agent credit assignment structure including local and global rewards, all agents are bonded to a main objective which is team-level optimization beside selfishly acquiring their own benefits [23]. SQL is used as a baseline method in this paper to evaluate our proposed learning method.

Algorithm 1: Swarm Q-Learning (SQL)

Input : Maximum number of episodes (T), Maximum number of steps in one episode (ϕ), Action set allowed for an agent to perform (A), Number of agents (K)
Output: Q-table (Q)
Initialize the states and the Q-table for an agent
for $t = 1$ **to** T **do**
  Set $\varphi \leftarrow 1$
  while $\varphi < \phi$ *and all agents are not at target states* **do**
    for $k = 1$ **to** K **do**
      if *agent* k *is not at target states* **then**
        Obtain the current state of agent k
        Use agent k Q table and ϵ-greedy algorithm to select action $a_l \in A$
        Observe the next state and reward r
        Update Q-values for agent k
      end
    end
    $\varphi \leftarrow \varphi + 1$
  end
  Each agent receives a bonus reward r_{bonus} depending on status of the targets
  Update the Q-values for the final state-action pairs
end

IV. Swarm Q-Learning with knowledge Sharing Within Environments

Phase 1 of the formation problem can be solved using different algorithms. However, to the best of our knowledge, these methods are either applicable when the environment is small or when they operate on each agent independently. Individual Q-Learning might not be a useful approach in this case because of the computational requirement of classic RL. Besides, this is attributed to the fact that target positions can only be occupied by one agent. Thus, lack of coordination among agents could incur extra cost at the team level; that is, agents keep searching for unoccupied targets.

In this section, we present our proposed method including two strategies to scale the algorithm to large search spaces while allowing for coordinated actions among the agents. The first strategy is decomposing the large search space into two sub-spaces, each corresponds to one sub-task. The second strategy is exchanging the learning knowledge among agents.

A. Search Space Decomposition Strategy

The decomposition of the large space into two sub-spaces can reduce the complexity of the searching problem. It allows for emergence of solutions that require different skill sets.

Assuming a square environment E with size $M \times M$, K agents, and K targets, each agent can perform L possible actions, the size of the search space D_E is represented by the following Equation 2:

$$D_E = L^{M \times M} \times (K!) \tag{2}$$

where $K!$ is the number of possible combinations between agents and targets.

Our proposed decomposition strategy is to split the original space into two sub-spaces as follows:

(1) **Sub-space 1**: navigation from starting area to the ROI where the agents need to form into the required formation, and (2) **Sub-space 2**: inside the ROI, each agent needs to occupy its position in the formation.

Let $m \times m$ be the dimensions of the ROI. We assume that m is much smaller than M ($m << M$). Then the size of sub-space 1 (D_{E1}) is:

$$D_{E1} = L^{(M-m)\times(M-m)} \times K \tag{3}$$

The size of the sub-space 2 (D_{E2}) is:

$$D_{E2} = L^{m \times m} \times (K!) \tag{4}$$

Clearly, the ratio between the size of the search space after decomposition and the original size decreases when the number of agents and targets K increases. However, there is a natural limit since if the two sub-spaces overlap, and with the assumptions that obstacles occupy the area in between, the problem reduces mainly to a problem in sub-space 2 alone.

1) Two Learning Processes: In this paper, our proposed method applies the Q-learning algorithm to each agent in two learning processes that correspond to the two sub-spaces. The first learning process is for agents to reach the boundary of the ROI. The second learning process aims at providing the agents with optimal policy to form the formation within the ROI.

The algorithm for the first learning process is classic Q-Learning. For the second learning process, we build K Q-matrices, each including state-action values guiding the agents toward one specific target. These Q-matrices are also obtained by applying classic Q-Learning. The K Q-matrices are shared among agents to find the closest target.

The second learning process is performed first by which we establish the ground-knowledge of how to reach targets within ROI. Then we train the agents on solving the combination of process 1 and process 2. Once one agent reaches one point (state) on the boundary of the ROI, it finds the closest target available based on a comparison among the Q-values of K Q-matrices at this state. Let Q_s^k be the set of Q-values of the k^{th} Q-matrix at state s. Then $Q_s^k = \{q_{(s,a_1)}^k, q_{(s,a_2)}^k, ..., q_{(s,a_L)}^k\}$, where a_l is the action l belonging to action set with L actions. The agent at state s chooses the Q-matrix corresponding to the maximum $q_{(s,a_l)}^k$ among all Q-values in K sets of $\{Q_s^1, Q_s^2, ..., Q_s^K\}$ which can be expressed in Equation 5.

$$k_s^* = argmax_k\{q_{(s,a_1)}^1, q_{(s,a_2)}^1, ..., q_{(s,a_l)}^k, ..., q_{(s,a_L)}^K\} \quad (5)$$

Where k_s^* is the index of the chosen Q-matrix at state s of the agent.

B. Knowledge Sharing Strategy

Because the first sub-problem forms a common goal to all agents, we can share the knowledge among agents in order to find the goal faster and avoid obstacles. Our proposed method adopts information exchange among the agents in learning process 1 of our decomposition strategy.

The procedure includes the selection of Q-values for the shared Q-matrix from Q-values of each individual agent after every episode. Because of the probabilistic processes, different agents acquire different experiences in each episode including exploring different routes to the ROI. After each episode, each agent has a policy to reach the ROI. This policy is represented by a Q-matrix, in which state-action pairs' Q-values encodes environmental information such as locations of obstacles and direction of goal. Based on this idea, we propose the Q-values sharing method as follows. After each episode, for each state-action pair from Q-matrices of all agents, we compare and select the maximum non-zero Q-value. All selected Q-values of all state-action pairs are then pooled into one value-shared Q-matrix that can be used as the base knowledge for every agent in the next episode.

$$Q_t^*(s,a) = max_k(Q_t^k(s,a)) \quad (\forall k, t, s, a) \quad (6)$$

The flow of the proposed SQL-SIE algorithm is shown in algorithm 2.

Algorithm 2: Swarm Q-Learning with knowledge Sharing wIthin Environments (SQL-SIE)

Input : Maximum number of episodes (T), Maximum number of steps in one episode (ϕ), Action set allowed for an agent to perform (A), Number of agents (K)
Output: Q-table (Q)
Initialize the states and the Q-table for an agent
for $t = 1$ **to** T **do**
  Set $\varphi \leftarrow 1$
  while $\varphi < \phi$ *and all agents are not at target states* **do**
    for $k = 1$ **to** K **do**
      if *agent* k *is not at target states* **then**
        Obtain the current state of agent k
        Use agent k Q table and ϵ-greedy algorithm to select action $a_l \in A$
        Observe the next state and reward r
        Update Q-values for agent k
      end
    end
    $\varphi \leftarrow \varphi + 1$
  end
  Each agent receives a bonus reward r_{bonus} depending on status of the targets
  Update the Q-values for the final state-action pairs
  Calculate Value-shared Q-matrix $Q_t^*(s,a)$ using Equation 6
  Copy $Q_t^*(s,a)$ to all Q-tables for all agents
end

The chosen maximum non-zeros Q-value at one state guides the search for avoiding the obstacles and aligning to goal direction. The higher Q-values represent the policy to obtain the higher reward, i.e avoiding obstacles and proceeding faster to the goal. Our knowledge sharing strategy takes advantage of a larger number of experiences of multi-agent exploration. This benefits the convergence speed of the learning method. To evaluate the impact of our knowledge sharing strategy alone, SQL-SIE is compared with a combination of an SQL algorithm described in Section III and our decomposition strategy without knowledge sharing (SQL-D).

V. Experiments

SQL-SIE is assessed through two experiments. The first experiment compares the performance of three approaches: (1) SQL, (2) SQL-D, and (3) SQL-SIE. The second experiment aims at analyzing the sensitivity of our knowledge sharing strategy to the exchanging frequency. The third experiment analysis the scalability of the SQL-SIE when we implement the method with different number of agents.

A. Experimental Setup

In the experiments, the environment is a 50×50 grid-world. There are 16 agents initialized at the starting area (lower left corner) and 16 targets initialized in ROI area (upper right

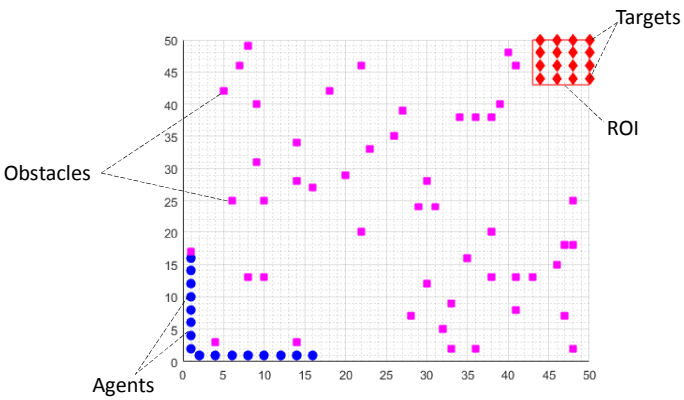

Fig. 2. Experimental Environment.

TABLE I
COMMON LEARNING PARAMETERS FOR EXPERIMENTS.

Parameter	Value
Maximum number of episodes (T)	10000
Maximum number of steps	1000
Learning Rate (α)	0.1
Initial epsilon (ϵ_0)	1.0
Epsilon decay	0.1%
Minimum value of epsilon	0.05
Discount factor (γ)	0.9

corner) in the shape of a box formation. ROI area is the square region at the top right corner of the environment (from $[43, 43]$ to $[50, 50]$.

Additionally, to increase the complexity of the problem, the obstacles are positioned randomly in the environment. The density of the obstacle accounts for 5% of the environment's area. Figure 2 illustrates the experimental environment.

Table I lists the common Q-Learning parameters for all methods in both experiments. The ϵ factor starts with 1 which triggers a full exploration mode. This factor gradually drops by 0.1% of the current value after each episode to exploit a greedier policy faster. A successful episode is the one with all targets occupied. If there are more than one target not occupied within 1000 steps, the episode is considered a failure. The number of actions of the last agent to finish the goal is considered as mission completion time for team. The three methods are compared on their ability to complete missions in the shortest possible time.

B. Learning in ROI

We perform RL in ROI (sub-task 2) to create target searching expertise and team-level optimal policy for the team members before learning the strategic conjugate of navigation (sub-task 1) and target searching (sub-task 2). However, due to the smaller size of the ROI compared to that of the whole environment, we set epsilon decay to a rate of 0.5% to accelerate the learning process. The rewarding scheme is similar to that of individual Q-learning with team reward.

Figure 3 shows the number of steps over training episodes for 4 targets as examples. The best policies for reaching target

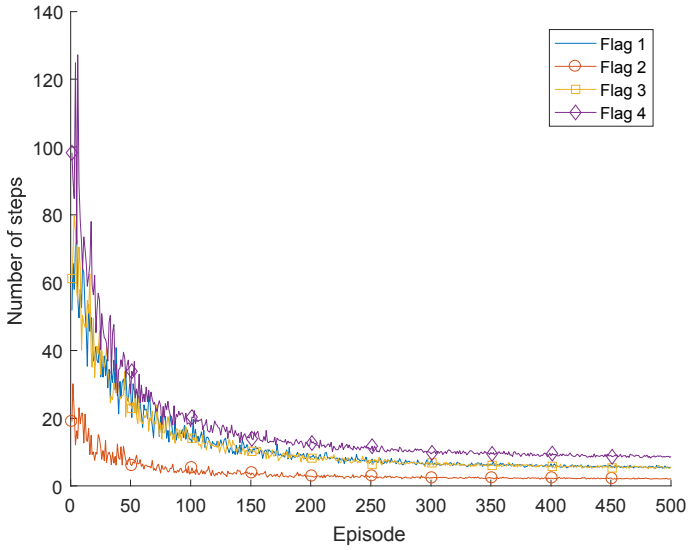

Fig. 3. Four curves representing average number of steps over 10 trials corresponding to four example targets when learning in ROI.

1, target 2, target 3, and target 4 found have the number of steps of 5, 2, 5, and 8 respectively after less than 500 episodes.

C. Experiment 1: Comparison among SQL, SQL-D, and SQL-SIE

1) Experimental Description: For SQL, the rewarding scheme is described as follows. If one agent has a collision with an obstacle, it obtains a negative reward of -1. Whenever one agent occupies a target, it receives an immediate reward of 10. If all agents reach all targets, they are all rewarded a positive team reward $r^+ = 10$. If not all targets are occupied, a negative team reward of $r^- = -1$ is received by the agent who reaches a target, and a greater negative reward $r^- = -2$ is gained by the agent who does not occupy any target at the end of the episode. Any collision to the obstacles rewards a -1.

On the other hand, our proposed methods, SQL-D and SQL-SIE, find the policy to navigate all agents to the ROI in sub-problem 1. When an agent reaches the ROI, it obtains a reward of $+10$. If an agent collides with an obstacle, it receives a negative reward of -1. During sub-problem 2, SQL-D and SQL-SIE use four Q-matrices obtained from learning in ROI before.

The performance metrics for this experiment are (1) the learning curve to compare convergence, (2) the number of steps of the last agent at convergence and (3) the difference between the number of steps of first arriving agent and last arriving agent to show their variations among learning.

2) Results: Figure 4 demonstrates that when using the SQL-D and SQL-SIE, the total number of team steps drops faster. The variation of the number of team steps in the case of using SQL is much higher than the other cases. The SQL-D and SQL-SIE are superior in the performance compared to SQL when the number of team steps reaches lowest numbers

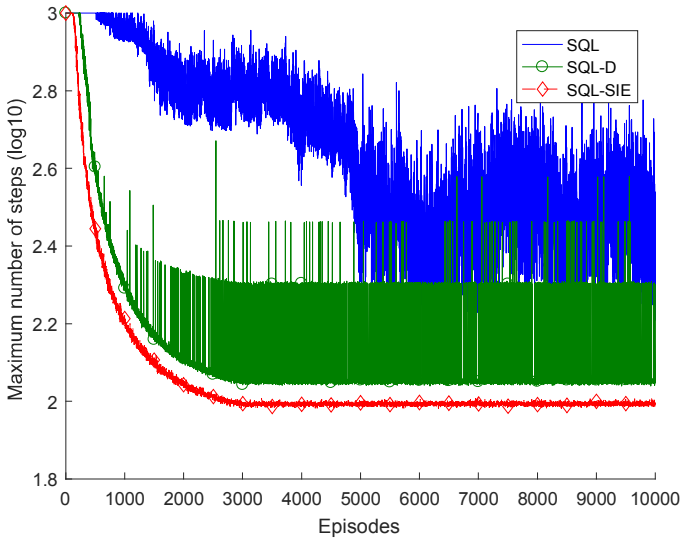

Fig. 4. A comparison among average maximum number of steps of all agents of three learning strategies over 10 trials.

TABLE II
COMPARISON OF THE AVERAGE NUMBER OF STEPS AMONG THREE METHODS AFTER EXPLORATION RATE REDUCES TO A LOWEST RATE OF 5% (AFTER EPISODE 3000).

Trials	**Methods**		
	SQL	SQL-D	SQL-SIE
Mean	378.0	128.0	98.5
Std	103.2	2.1	0.2

of around 111 and 95 steps from episode 3000 (reach the minimum degree of exploration (5%)) on average respectively. However, the behaviors of learning curves show that only SQL-SIE achieves a stability at the end of training period while SQL-D's performance still fluctuates significantly. SQL-SIE is the only method to achieve a convergence of Q-learning within the specified training time of the experiment.

In all learning methods, we set a minimum values of epsilon at 0.05 to allow 5% of chance for exploration at the end of the training period. Therefore, after 3000 episodes, the agents follow near fully exploitation mode. Table II shows the average number of steps after the minimum exploration rate has been reached. The results demonstrate that the decomposition of the search space helps in improving the speed and stability of learning.

SQL consistently generates strategies that takes longer to reach the goal as indicated by the largest mean number of steps among all methods and the largest standard deviation as well. SQL-D and SQL-SIE are more efficient as demonstrated by the small mean number of steps. They are also more stable as demonstrated by the small standard deviation. Out of the three methods, SQL-SIE generates most efficient solutions with better stability. This superior performance of SQL-SIE is statistically significant ($p = 9.15 \times 10^{-8}$ and 7.17×10^{-20} when comparing to SQL and SQL-D respectively at a significant level of 0.05). The employment of knowledge sharing among agents allows faster exploration, achieves a lower number of steps at similar episodes, and stabilizes the performance of the learning method.

TABLE III
MEAN DIFFERENCE BETWEEN THE NUMBER OF STEPS OF FIRST ARRIVING AND LAST ARRIVING AGENTS AFTER EPISODE 3000.

Trials	**Methods**		
	SQL	SQL-D	SQL-SIE
Mean	298.2	41.1	21.3
Std	103.4	2.5	0.3

On knowledge sharing strategy, further analysis on the number of steps of individual agents in three cases suggests the reason for the high variation and the sub-optimal solution of the SQL and SQL-D. Table III illustrates the mean difference of the number of steps over exploitation phase between the first arriving and the last arriving agents.

In SQL, in some cases, only some agents reach a convergence while other still cannot find the remaining target locations. The difference between the number of steps of the first arriving agent and the last arriving one is as high as 298.2 on average. This results in the very high variation of the number of team's step. In this case, the algorithm fails to reach a stabilized policy for the whole team. The SQL-D and SQL-SIE methods reduce the difference between the number of step taken by the first agent who completes the task and the last one to around 41.1 and 21.3 steps on average. Because the number of steps of the final agent who reaches the goal is considered as the number of team's steps, in the learning strategies with no knowledge sharing (SQL and SQL-D), the differences in completion time among agent are extended. The variation is statistically significantly lower when sharing the knowledge among agent with SQL-SIE ($p = 1.08 \times 10^{-7}$ and 2.08×10^{-15} when comparing to SQL and SQL-D respectively at a significant level of 0.05). It is demonstrated that the information exchange among agents helps reduce the variation among the number of steps of all agents, and hence, reduce the completion time.

D. Experiment 2: Sensitivity Analysis of Knowledge Sharing Frequency

1) Experimental Description: To analyze the sensitivity of the knowledge sharing strategy, we use SQL-SIE on different updating frequencies of the shared Q-matrix. We compare the behaviour of learning curves when using different knowledge sharing periods: every 1 episode, every 20 episodes, and every 50 episodes.

2) Results: Figure 5 shows that the more frequently the shared Q-matrix is updated, the lower the total number of learning steps overall learning process. These properties can be demonstrated by the area under the learning curve. When updating the shared Q-matrix as frequently as every one episode, the area under the learning curve is much smaller than the longer periods. For this large environment, higher sharing frequency improves team performance and stability when exploring and exploiting the learned knowledge. On the

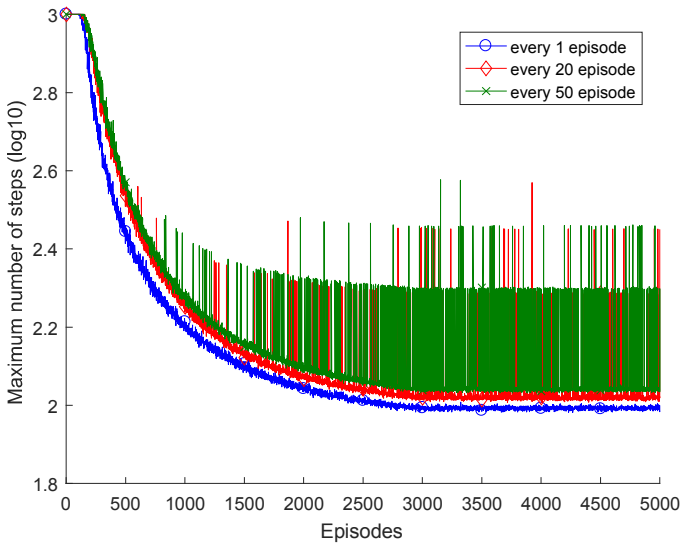

Fig. 5. A comparison among the learning behaviors of different knowledge sharing frequency in SQL-SIE.

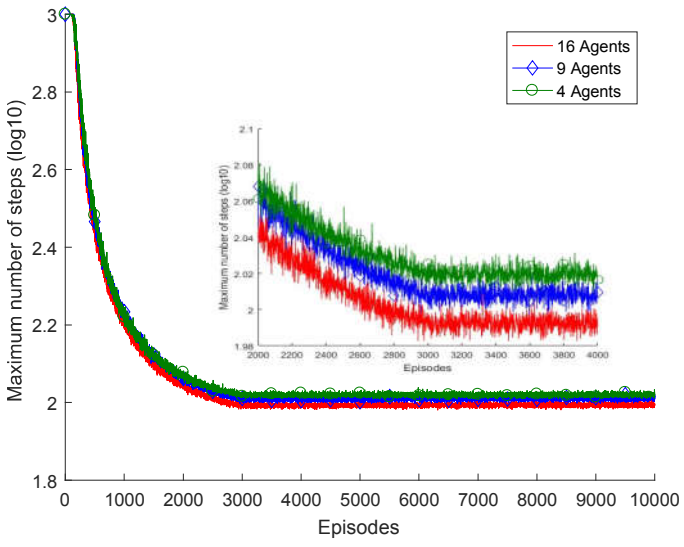

Fig. 6. A comparison among the learning behaviours of different knowledge sharing frequency in SQL-SIE (Subfigure: plot from 2000 to 4000 episodes).

other hand, lower frequencies of knowledge sharing may result in instability and sub-optimal policies. Besides, the higher frequency of knowledge sharing does speed up the learning due to the fact that the number of steps reduces faster.

E. Scalability of SQL-SIE

1) Experimental Description: The Equation 2 shows that as the number of agents and targets increases, the size of the search space increases factorially. Therefore, the performance of our search space decomposition and knowledge sharing strategies is enhanced with an increasing number of agents in a same size of environment. In this experiment, we test our hypothesis above by implementing SQL-SIE method on three settings: 4 agents-targets, 9 agents-targets, and 16 agents-target positioned in the 50×50 environment. The environment has the same structure as the environment in Experiment 1 and 2.

2) Results: Figure 6 illustrates the learning behaviors of agents using SQL-SIE in three settings. The results suggest that increasing number of agents would help accelerate convergence in sub-space 1 because there is a common goal and agents collaborate to achieve the goal. The use of our knowledge sharing strategy allows the stability of the learning method.

As demonstrated from Figure 6, when the number of agents increases, the maximum number of steps of the whole agent team decreases. Despite the fact that the performance increases slightly with an increasing number of agents, the convergence time of all settings is similar one another. Thus, it is concluded that our knowledge sharing strategy is scalable when the number of agents increases.

VI. Conclusion

This paper proposes a search-space decomposition and knowledge sharing strategies to apply in a multi-agent formation initiation problem. Q-Learning is used in the proposed method to learn the policy of individual agents for achieving the lowest number of actions of the whole agent team operating in a large and complex environments.

Our search-space decomposition strategy effectively reduces the size of the search space. The results show that the decomposition of search space helps improve the learning stability and the learning speed. Besides, the average number of steps of whole multi-agents team when applying search space decomposition is much lower than the average number of steps in SQL, the baseline method in research [22].

Further investigation into our knowledge sharing strategy suggests a significant suppression of variation in learning. When comparing with no knowledge sharing strategy (SQL-D), the method with knowledge sharing (SQL-SIE) provides significantly higher learning speed and the learning stability. Additionally, the average number of team's steps decreases remarkably.

In order to evaluate the sensitivity of our knowledge sharing strategy, different information exchange frequencies have been examined. The behaviors of learning curves shows that sharing information in a shorter period of time leads to a better performance.

Last but not least, our method has been tested on different number of agents in the same size of environment. The numerical results support that our proposed method has an improvement of learning performance while no significant increase in convergence time. This demonstrates that our knowledge sharing strategy has scalability property.

Future work will aim at solving a composite problem of initializing a predefined formation and maintaining that formation during operations in a large search space. Non-tabular approaches such as deep Q-networks might be integrated with a knowledge sharing strategy to fuse different sources of input

for state representation and to generalize the team's policy on similar situations.

ACKNOWLEDGEMENT

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA2386-17-1-4054.

REFERENCES

[1] H. Oh, A. R. Shirazi, C. Sun, and Y. Jin, "Bio-inspired self-organising multi-robot pattern formation: A review," *Robotics and Autonomous Systems*, vol. 91, pp. 83–100, 2017.

[2] D. J. Stilwell and B. E. Bishop, "Platoons of underwater vehicles," *IEEE control systems*, vol. 20, no. 6, pp. 45–52, 2000.

[3] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen, "A survey of spacecraft formation flying guidance and control. part ii: control," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 4. IEEE, 2004, pp. 2976–2985.

[4] A. Serrani, "Robust coordinated control of satellite formations subject to gravity perturbations," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 1. IEEE, 2003, pp. 302–307.

[5] A. Robertson, T. Corazzini, and J. P. How, "Formation sensing and control technologies for a separated spacecraft interferometer," in *American Control Conference, 1998. Proceedings of the 1998*, vol. 3. IEEE, 1998, pp. 1574–1579.

[6] M. Martin, P. Klupar, S. Kilberg, and J. Winter, "Techsat 21 and revolutionizing space missions using microsatellites," 2001.

[7] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," *IEEE Transactions on control systems technology*, vol. 9, no. 6, pp. 777–790, 2001.

[8] J. L. Baxter, E. Burke, J. M. Garibaldi, and M. Norman, "Multi-robot search and rescue: A potential field based approach," in *Autonomous robots and agents*. Springer, 2007, pp. 9–16.

[9] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[10] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998.

[11] J. C. Derenick and J. R. Spletzer, "Convex optimization strategies for coordinating large-scale robot formations," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1252–1259, 2007.

[12] W. Li and W. Shen, "Swarm behavior control of mobile multi-robots with wireless sensor networks," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1398–1407, 2011.

[13] D. Xu, X. Zhang, Z. Zhu, C. Chen, and P. Yang, "Behavior-based formation control of swarm robots," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[14] A. Guillet, R. Lenain, B. Thuilot, and V. Rousseau, "Formation control of agricultural mobile robots: A bidirectional weighted constraints approach," *Journal of Field Robotics*, 2017.

[15] A. Sen, S. R. Sahoo, and M. Kothari, "Cooperative formation control strategy in heterogeneous network with bounded acceleration," in *Control Conference (ICC), 2017 Indian*. IEEE, 2017, pp. 344–349.

[16] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*, 2008.

[17] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[18] R. Guerraoui, H. Hendrikx, and A. Maurer, "Dynamic safe interruptibility for decentralized multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 129–139.

[19] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 464–473.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[21] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[22] H. Iima and Y. Kuroe, "Swarm reinforcement learning method for a multi-robot formation problem," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2298–2303.

[23] P. Mannion, K. Mason, S. Devlin, J. Duggan, and E. Howley, "Dynamic economic emissions dispatch optimisation using multi-agent reinforcement learning," in *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2016)*, 2016.