AI Communications 0 (0) 1
IOS Press

Multi-agent Reinforcement Learning Improvement in a Dynamic Environment Using Knowledge Transfer

Mahnoosh Mahdavimoghaddam [a], Amin Nikanjam [a,c] and Monireh Abdoos [b,*]
[a] *Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran*
E-mails: mahnooshmahdavi2012@gmail.com, nikanjam@kntu.ac.ir
[b] *Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran*
E-mail: m_abdoos@sbu.ac.ir
[c] *SWAT Lab., Polytechnique Montréal, Quebec, Canada*
E-mail: amin.nikanjam@polymtl.ca

Abstract. Cooperative multi-agent systems are being widely used in variety of areas. Interaction between agents would bring positive points, including reducing costs of operating, high scalability, and facilitating parallel processing. These systems pave the way for handling large-scale, unknown, and dynamic environments. However, learning in these environments has become a prominent challenge in different applications. These challenges include the effect of size of search space on learning time, inappropriate cooperation among agents, and the lack of proper coordination among agents' decisions. Moreover, reinforcement learning algorithms may suffer from long time of convergence in these problems. In this paper, a communication framework using knowledge transfer concepts is introduced to address such challenges in the herding problem with large state space. To handle the problems of convergence, knowledge transfer has been utilized that can significantly increase the efficiency of reinforcement learning algorithms. Coordination between the agents is carried out through a head agent in each group of agents and a coordinator agent respectively. The results demonstrate that this framework could indeed enhance the speed of learning and reduce convergence time.

Keywords: Cooperative multi-agent systems, Dynamic environments, Reinforcement learning, Knowledge transfer

1. Introduction

These days, many problems can be solved using multi-agent systems, and cooperation among them. Using parallel processing in multi-agent systems speeds up the applications. Agents can cooperate together to obtain a common goal, or compete for conflicting aims. Agents at times exchange or negotiate on a topic[1–3].Reinforcement learning (RL) is an effective solution as a famous machine-learning tool for learning in multi-agent systems, which is employed to come up against complex decision-making problems in uncertain environments. The RL makes an agent enable to progressively learn a sequence of actions to achieve the desired goals [4].
While applying RL, the environment in which the agent carries out an action, provides scalar feedback which is called reinforcement/reward signal, and as a result of the action, the agent transits from a state (current state) to a new one. During the process of interaction with the environment, the agent investigates the environment and accumulates experiences or knowledge considering the environment. While employing the accumulated experiences the agent learns what is the best sequence of action from any situation/state to achieve the desired objective. In the model-based RL approaches such as Q-learning, the agent strives to rebuild a virtual model of the environment

*Corresponding author. E-mail: m_abdoos@sbu.ac.ir.

which includes the reward function and also the transition function, this function encodes how an action from the current state is mapped to the next state. Therefore, the procedure of achieving experiences and reconstructing a virtual model of the environment in an uncertain environment is exorbitant and time-consuming, and the gathered experiences must be efficiently utilized [5].
This study was conducted in a grid-like dynamically changing environment where the agent carries out an action based on the partial perceptions of the environment to achieve the target and learn based on the received feedback. Herein, not only the problem of prolonged convergence time due to large search space is studied, but also the lack of proper coordination among agents' decisions as a result of inefficient cooperation is investigated.
The contribution of this paper is summarized as:

- A new approach is proposed to reduce the size of the state space,
- A new model-learning based algorithm is indicated in order to enhance cooperation among members of a group,
- A new communication framework is presented in order to improve coordination among agents,
- An efficient knowledge-sharing mechanism is introduced to accelerate the learning process.

The rest of the paper is organized as follows: In Section 2, a brief background on related works is presented. The problem considered in this paper is detailed in Section 3. The proposed model learning method along with knowledge-sharing algorithms are presented in Section 4 and Section 5. Section 6 is devoted to introducing some heuristic techniques which are employed to check the results. In Section 7, the simulation results for the proposed method are presented. Finally, in Section 9, we present our conclusion.

2. Literature review

There have been many studies on multi-agent reinforcement learning (MARL). The MARL is a challenging research area employed in various fields, including robotics, resource allocation, traffic signal control, shepherding problem, and many others [1–3].
Although RL algorithms have been highly successful in multi-agent systems, the large amount of data and lengthy exploration times that such algorithms require make them intractable for many real world robotic areas. Numerous studies have tried to address this issue such as in [6, 7]. [6] introduces a reinforcement learning and curriculum transfer learning method to regulate multiple units in StarCraft micromanagement. The principal focus of [7] is on human guidance to reduce the number of explored states and the time of learning, under the condition that each guidance message limits the set of options for action selection to a small number of actions.
Some researchers applied a knowledge-sharing algorithm to reduce the effort required for exploration and to decrease the time of learning in a large state space environment [8]. In [8] a model-based RL scheme was presented wherein the accumulated experiences are not only used for policy learning but also used for model learning to estimate the environment.
The shepherding problem was selected here and investigated. Bayazit, Lien and Amato [9] proposed a rule-based roadmap regulation system to generate homing, exploring, and shepherding swarm behaviors. Each roadmap comprised of a series of nodes joined with edges.
A simple algorithm introduced by Miki and Nakamura [10] was utilised to simulate a herding task. The authors present four rules for the behavior of the swarm: cohesion to the nearest neighbor, separation to avoid collisions with obstacles or other swarm agents, run away from the shepherd(s), and random action making random stochastic movements. Similarly, the shepherd was controlled by four regulations: guidance of the flock in an objective direction, flock creating to push scattered sheep back to the group, keeping a particular distance from the flock to avoid splitting it, and cooperation to avoid shepherds overlapping. The results illustrated the algorithms considerably repeated herding behaviors [11].
A special shepherding control system was created by Razali, Meng and Yang [12], driven by the theory of immune networks, permitting a distributed control system, and is able to adjust actively to an environment. The memory-based immune system carried out as an analogy for the shepherding problem, where obstacles are shown as antigens, robots as B-cells, and actions as antibodies. The results of the study, while not argued in more detail, did show that the solution was not efficacious at shepherding robotic swarm agents. A follow-up research by Razali, Meng

and Yang [13], however, developed a more in-depth discussion of their method. [13] introduces a refinement of the memory-based immune network that improves a robot's action-selection procedure.
Strombom et al. [14] designed a simple heuristic to imitate sheep behaviors, utilising just a single shepherd, which closely matched that of [10], where the two-dimensional (2D) simulations comprised of the same behavioral principles. Differences consisted of swarm attraction to the center of mass (CM) of their local neighborhood (in the place of just a single neighbor), and the solo-shepherd not requiring collaboration with another shepherd, and the terms guidance and flock making replaced with driving and collecting [11].
Pierson and Schwager [15] applied the same multi-shepherd regulation strategy, where the shepherds steered a swarm agent by spreading themselves along an arc centered on the agent, using a single continuous control law which negated the requirement for behavioral switching.
Fujioka and Hayashi [16] assessed a substitute shepherding control behavior called V-formation control, in which the shepherd cycles among three positions along an arc extending out from a swarm's CM, centered behind, and rotated slightly to the left and right ($< 90\circ$ from the center position), making a V-shaped trajectory.
Another algorithm, presented by Lee and Kim [17], applied an alternative set of swarm and shepherd behaviors to come up against a shepherding problem. If swarm agents are within a specific distance from each other, then they endeavor to minimize the distance between themselves. The agents then try to move to the center of their swarm neighborhood, while striving to match their neighbors' velocities. The swarm agents keep away from collisions with both obstacles and the shepherds. The shepherds collect scattered agents into a single swarm, then guide them towards a goal destination. Lee and Kim, then, included patrolling (keeping the swarm at a fixed position) and covering (steering the swarm to multiple goals) shepherd behaviors into their simulations. An extra methodology and discussion was added, and their algorithm was shown to be useful at independently regulating the swarm [11].
In 2018, Hoshi et al. performed two studies [18], [19], which elaborated on the work of Strombom et al. [14]. The first in [18] tested the effects of differing the step size per time step of the shepherd and swarm agents on the shepherding task. The task was carried out better once the shepherd was able to move faster in comparison with a swarm which moved at the same pace. In the next study [6], the shepherd and swarm agent's movement and influential force vectors were added to the third dimension [11]. Over 30 years, a large body of research has been accumulated on the use of shepherding as a swarm control technique. Among all the research that has been done in the field of herding, a research gap in the field of reducing the size of the state space and improving the coordination between the agents is quite visible. Reducing the number of explored states and the learning time by restricting the set of choices for action selection to a small number of actions inevitably reduces the independence of the agent. Using the concepts of knowledge transfer for increasing the speed of learning and reducing the time of convergence can have a significant impact on the learning algorithms used in the mentioned research.

3. Problem description

Multi-agent systems developed in the context of the Multi-Agent Programming Contest were reported. The contest task includes creating a multi-agent system (MAS) to solve a cooperative task in a grid-like dynamically changing world where agents in a team are able to move from one cell to a neighboring one in each time step. The scope of the competition includes two teams of agents collecting cows scattered around the world and herding them into a corral. The actual implementation of the system is done in the JIAC programming language, which is a Java extension. All development and running of agents were performed on standard PC hardware, JIAC provides a Distributed Communications Infrastructure (DCI), where each agent is responsible for its processing and functionality. Therefore, decentralization makes robotic systems far easier to make and program, as well as presenting robustness, flexibility, and scalability into the multi-robot system. All agents receive XML messages from the game server corresponding to the various game phases: the beginning of a game (sim-start); the end of the game (sim-end); the end of a simulation run of a lot of games (bye); and within a game, where the game server constantly requests a next move (request-action). the system sends action demands to the game server; either to move a player to an adjacent cell, or a "skip" action, i.e., no movement (the game server additionally implements a response timeout, defaulting the action to be a skip action if the system fails to issue its action choice during the time limit). The following scenarios are repeated for all agents in each timestamp:

- Understanding the grid (finding cows, boundaries, other agents, and corrals),
- Collecting cows and pushing them into our corral,
- Engaging with the opponent team (Stealing the opponent's collected cluster in the middle of their path).

To solve the above task, an appropriate communication framework (using JIAC programming language) was introduced, and in order to train agents, RL as a popular machine-learning tool employed to tackle complex decision-making problems in uncertain environments were used. The RL enables an agent to progressively learn a sequence of actions to achieve the desired objective [4].

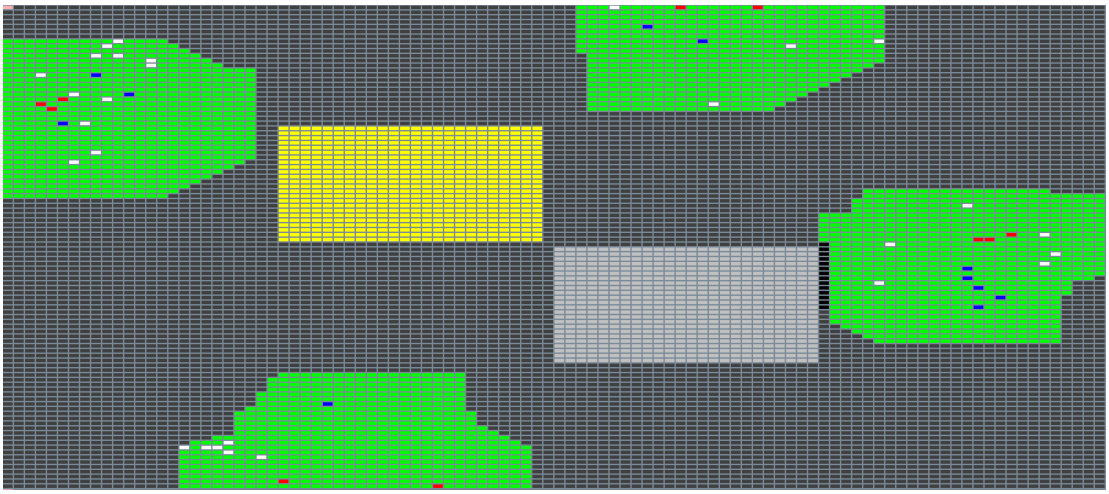

Fig. 1. Graphic space of the shepherding problem

As it can be seen in Fig. 1, the green, yellow, light, gray, and dark areas are tree, corral, opponent's corral, and unknown, respectively. The white color represents the cows and the two colors red and green demonstrate the two rival teams.

4. The proposed method

In this Section, first a general route map is introduced and then the details are explained. In the exploring phase, the player agents randomly select an action from the 8 available actions to be transferred to the adjacent cell, then in the next step, they update their learning Q-table according to the changes made. In the exploitation phase, player agents use their learning Q-tables to select suitable actions. In the following solutions to reduce the size of the state space are introduced, and the knowledge transfer algorithm among agents will be explained.

4.1. System roles

Once all of the agents have been identified, a special focus is placed on stating how these agents interact with each other to provide the required system performance. In the communication framework, the coordinator agent was introduced as an internal agent to the system, as it does not interact with other agents in the environment. The system consisted of a single coordinator agent and several player agents. The former representative is responsible for team-level support functionality. Recent agents are those playing the actual game on the game server. The game server (MASSIM server) interacts with player agents via TCP/IP sockets employing an XML based messaging protocol. Within a game, the server issues messages to agents explaining their current perceptions and permitting every agent to submit an action during a certain period of time. The MASSIM server interaction is described as a perceived + act cycle, where the agent constantly (once every time step) achieves a percept and submit an action.

4.1.1. Coordinator agent

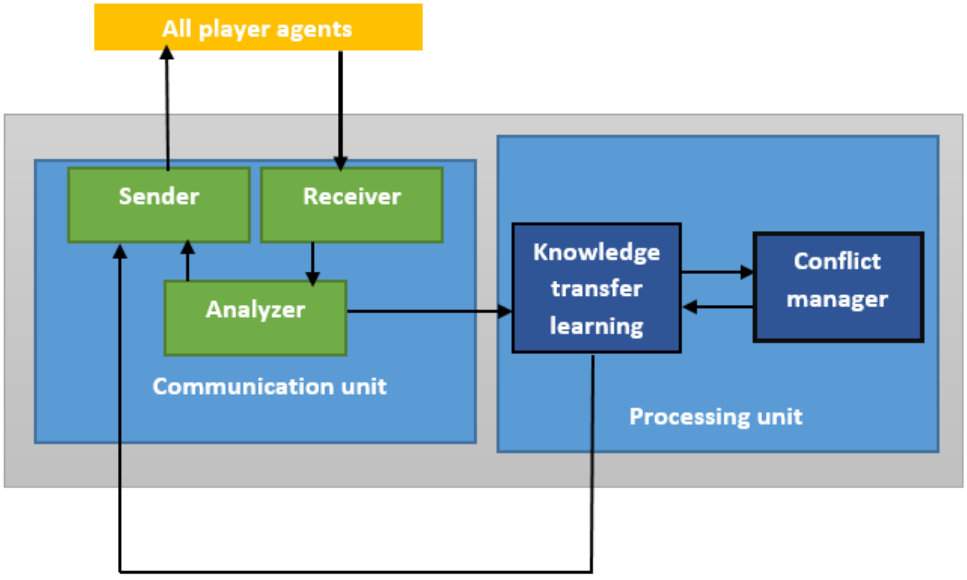

Fig. 2. Structure of coordinator agent

At the beginning of the simulation, the server picks two teams of agents to participate in the simulation. The simulation starts by notifying the corresponding agents about the details of the starting simulation. This notification is done by sending the SIM-START message. This message contains the upper, lower, left, and right border of the corral (numeric) . According to the Fig. 2, in the first step of the game, all player agents send their coordinates to the coordinator agent, as this is not an obstacle-free environment the coordinator agent, after receiving the coordinates of all the player agents in the Analyzer unit, is responsible for finding the player agent that is closer to the middle of the corral and sends it a message in order to makes it responsible for identifying valid corral's entrances. After all the entrances have been identified, this agent sends the array of valid coordinates to the coordinator agent, and this agent sends this array to all player agents. The second function of this agent is related to the concept of knowledge transfer, after determining the coordinates of all the entrances of corral, at the end of each time step, all player agents send their Q-tables to the coordinator agent. The receiving unit, after receiving the Q-tables, sends them to the analyzer unit. If the received message was a table type, the analyzer unit will send that table to the knowledge transfer unit. In the conflict manager unit, the knowledge of all similar Q-tables (each behavior has a separate Q-table) is aggregated into a table and this table is returned to the knowledge transfer unit to be sent to all agents. This will be explained in more depth in Section 5. In Fig. 3 the flow chart of coordinator agent could be seen.

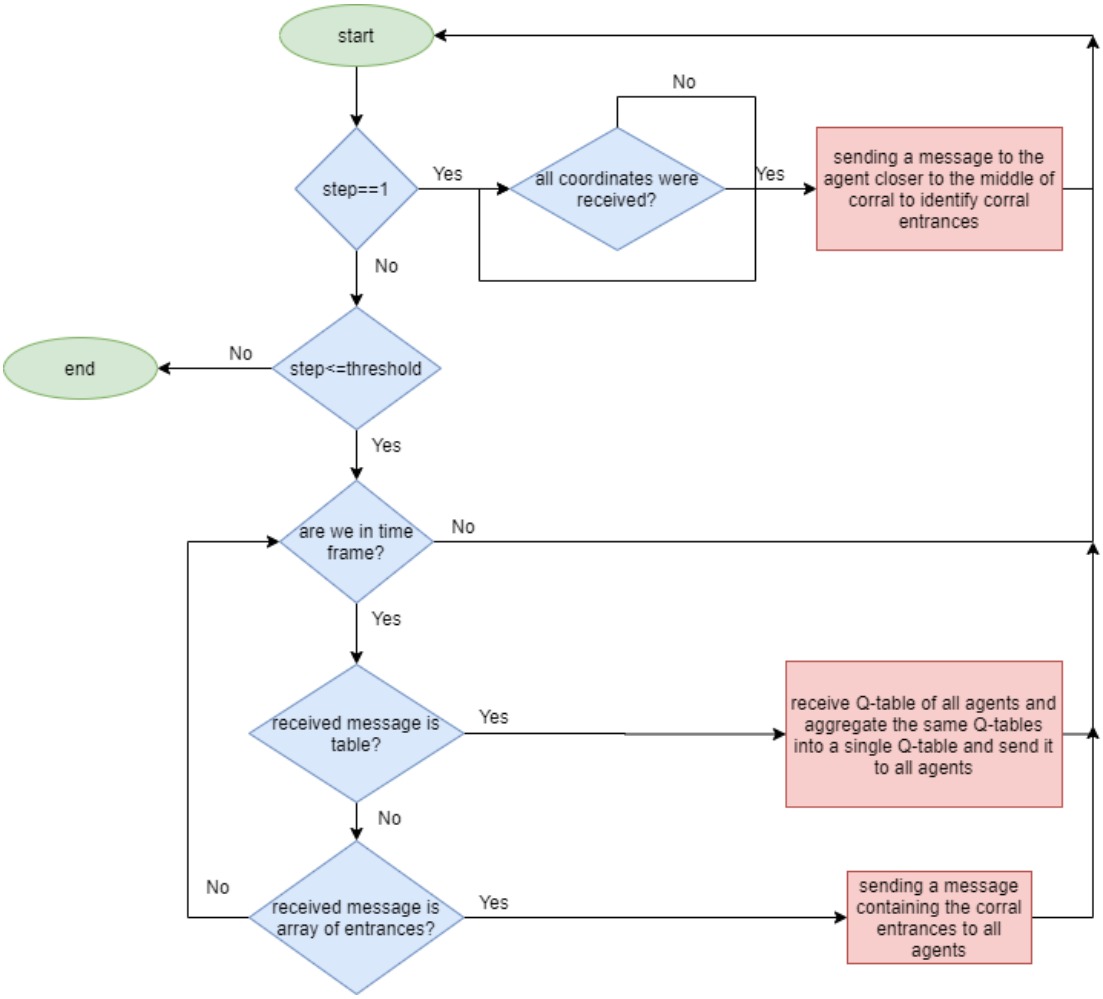

Fig. 3. Flowchart representing the coordinator agent

4.1.2. Player agent

At the beginning, all player agents are in the position of waiting to receive a message from the coordinator agent to start searching around the corral and select random action before receiving that message. After the agent closer to the middle of the corral receives the message, the other agents continue to select the random action. However, in the action selection unit, this agent selects actions according to the upper, lower, left, and right border of the corral(numeric) that have been received from SIM-START message, to identify obstacle-free entrances of corral. After all the valid entrances have been found, the array of these coordinates will be sent to the coordinator agent through the communication unit, and coordinator agent sends this array to all the agents. After specifying all the entrances for all agents, the process of training the agents will start. In each time step, the nearest corral entrance to the collected cluster as the target was considered. Fig. 4 shows the structure of the player agents. Observations received from the environment are delivered to the percept analysis unit. The player agent will then follow the following steps.

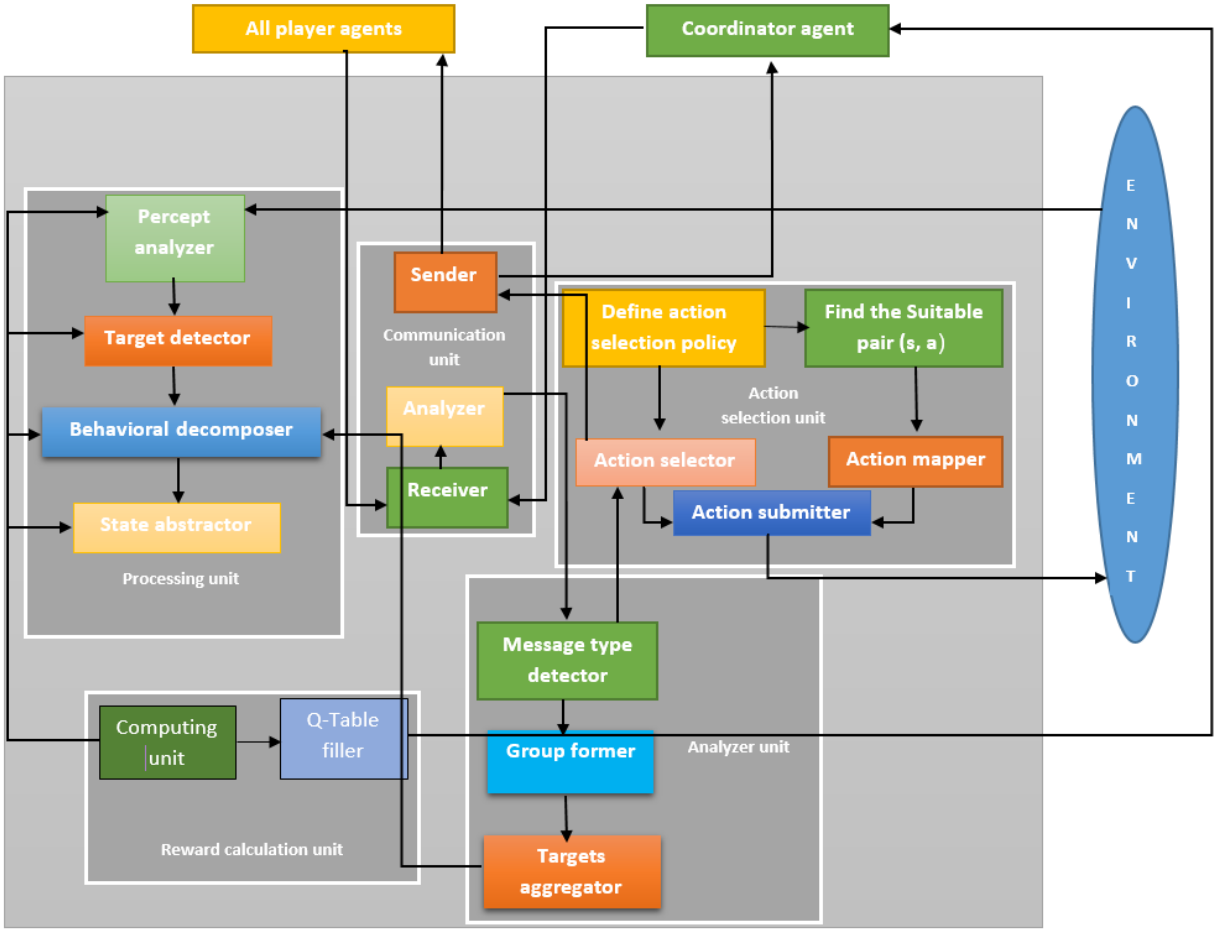

Fig. 4. Structure of player agent

4.2. Tasks

The player agent type will represent each agent registered in the game server and playing the actual game. These are the agents that can perceive and act in the game server. Because of that, these agents need to be able to fulfill the system roles that have to do with carrying on activities in the game. A coordinator agent is an agent that is able to fulfill the coordination functionality, which amounts to facilitating team behavior within a group of player agents.

4.2.1. Target detector

After receiving the information of four squares eight by eight around it (the line of sight is 8), each agent performs the following tasks:

- Sheep clustering based on proximity threshold and herd formation,
- Select a more populous herd as the current target.

If the agent receives a request for cooperation and is willing to cooperate, the targets of other members will be added to its targets in the target aggregator unit.

4.2.2. Behavioral decomposer

As mentioned earlier in each time step, the nearest corral entrance to the collected cluster as the target was considered and then the environment was divided into two parts, facing the target and back with the target. According to Fig. 5, if the target point is on the right border of the corral. The environment was divided into these two parts, A and B.

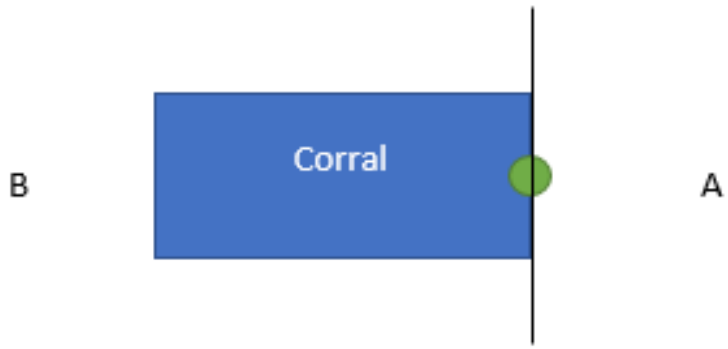

Fig. 5. Segmentation of the environment around the corral

Inspired by the model introduced by Strömbom et al. [14], shepherding requires two basic behavioral building blocks and the switching between them, namely, collecting and driving. Based on this definition, the following behaviors are introduced:

- Solo herding: If the single agent and the collected cluster are in zone A and the angle between the agent and the collected cluster and the target point is greater than a threshold (greater than 120±10).
- Group herding: If the multi-agent and the collected cluster are in zone A and the angle between the agent and the collected cluster and the target point is greater than a threshold (greater than 120±10).
- Solo following: If the single agent and the collected cluster are in zone A and the angle between the agent and the collected cluster and the target point is less than a threshold (less than 120±10).
- Group following: If the multi-agent and the collected cluster are in zone A and the angle between the agent and the collected cluster and the target point is less than a threshold (less than 120±10).
- Solo transferring: If the single agent and the collected cluster are in zone B.
- Group Transferring: If the multi-agent and the collected cluster are in zone B.

The shepherding behavior is deconstructed into switching, solo or group herding, solo or group following, or solo or group transferring with each of the six latter behaviors individually trained on separate Q-tables.

4.2.3. State Abstractor

As reinforcement learning depends on a state of the agent in each time step and the problem is algorithmically difficult due to the complexity of its search space, for this reason state abstraction was used. By using state abstraction, irrelevant features of state space can be ignored and the size of state space can be reduced.
According to Fig 6, when agent i is alone the state in each time step could be described as in Eq.1:

$$S_{a_i} \simeq \left(\left[\frac{D_{a_i t}}{d} \right], \left[\frac{D_{a_i c}}{d} \right], \left[\frac{\alpha}{a} \right] \right) \tag{1}$$

In Eq. 1, R is the size of each side of the environment as well as $1 \leqslant d \leqslant R\sqrt{2}$, $1 \leqslant \alpha \leqslant 180$

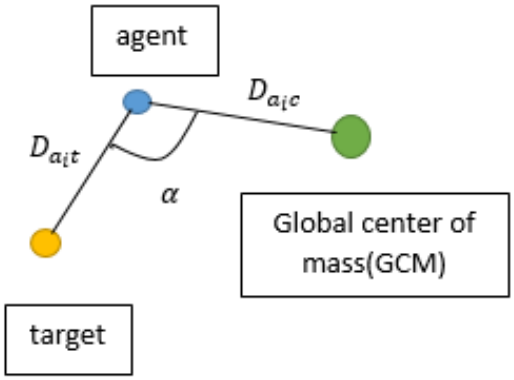

Fig. 6. Illustration of one-agent state

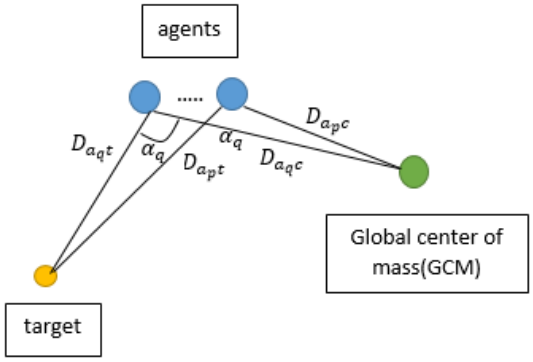

Fig. 7. Illustration of multi-agent state

As you can see in Fig. 7, when the number of agents pursuing the goal is more than one the state could be described as in Eq.2:

$$S_q = ... = S_p \simeq \left(\left[\frac{\sum_{i=q}^{p} D_{a_it}}{M \times d} \right], \left[\frac{\sum_{i=q}^{p} D_{a_ic}}{M \times d} \right], \left[\frac{\sum_{i=q}^{p} \alpha_i}{M \times a} \right] \right) \tag{2}$$

In Eq.2, R is the size of each side of the environment as well as $1 \leqslant d \leqslant R\sqrt{2}, 1 \leqslant \alpha \leqslant 180, i \subseteq \{q, \ldots, p\}$.

M is the number of agents that are in the same group. By removing features unrelated to the joint action and using the concept of action abstraction, the joint action could be considered as the sum of the actions of all the agents of a group.

Since the environment is continuous, through factors d and a, many states are similar to each other and can be stored as one state, thus significantly reducing the size of the search space.

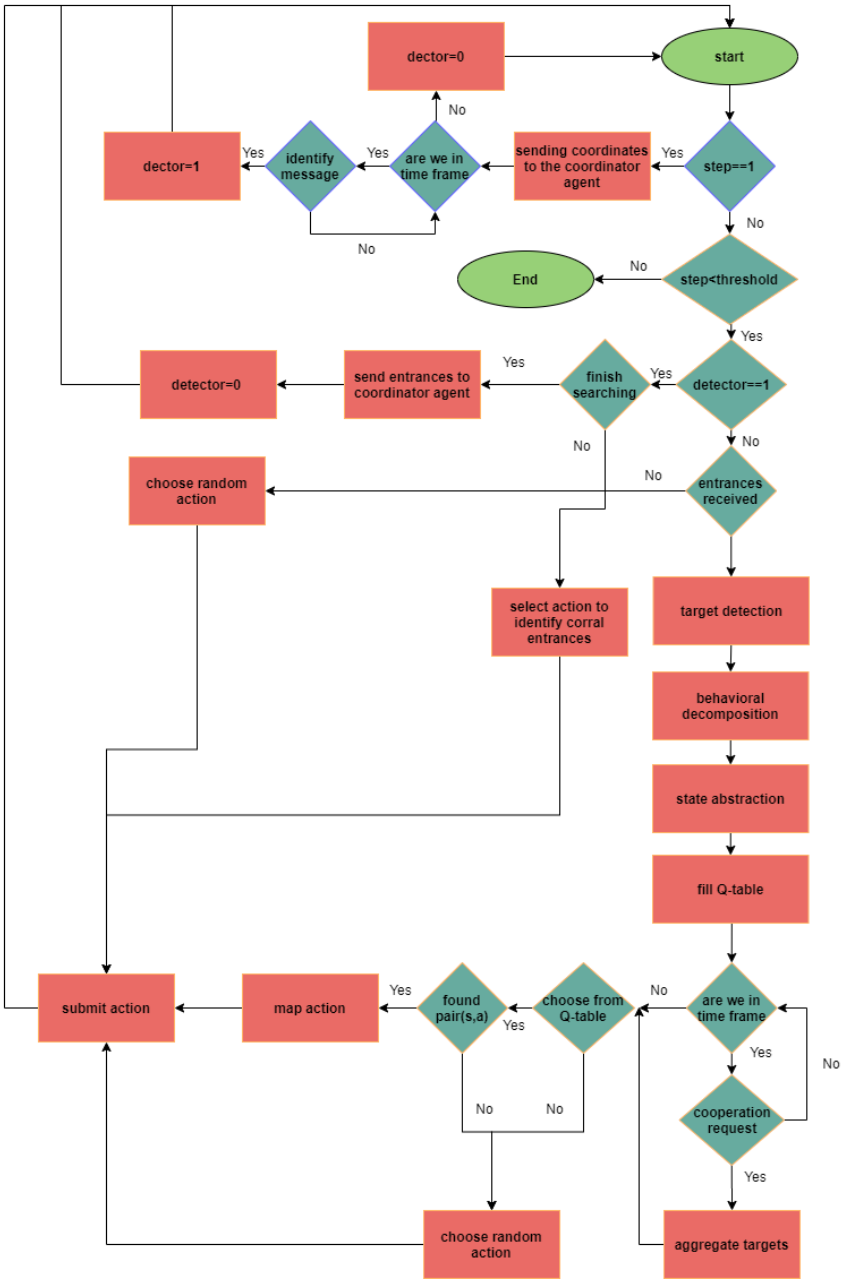

Fig. 8. Flowchart representing player agent

After specifying the state of the player agent, in the action selection unit, based on action selection policy, either an action is selected randomly or is selected according to the appropriate pair(s, a) in the similar behavior Q- table and finally submitted. At the next time step, in the reward calculation unit, the Q-table corresponding to the previous behavior of the agent will be updated according to the reward function and the changes in the perceptions. Previous state, behavior, position of the target and the new observations are all available in state abstractor, behavioral decomposer, target detector and percept analyzer units, respectively. In Fig. 8 functionality of player agent has been demonstrated.

5. Knowledge transfer

In the cooperative knowledge sharing approach, each agents' policy/model is shared with the others, and an agent has two important tasks [5]:

- Effectively combine the newly obtained knowledge from the other agents with the knowledge gained using its own experience,
- Effectively map the shared knowledge to the existing state.

5.1. *Effectively combine*

As can be seen in Fig. 3, at the end of each time step, all agents send their Q-tables to the coordinator agent. The receiving unit, after receiving the Q-tables, sends them to the analyzer unit. If the received message was a table type, the analyzer unit will send that table to the knowledge transfer unit. In the conflict manager unit, the knowledge of all similar Q-tables (each behavior has a separate Q-table) is aggregated into a table and sent to all agents.
In conflict manager unit each Q-value in the new table could be described mathematically as in Eq.3:

$$Q_{SH}(s,a) = \frac{\sum_{i=1}^{N} Q_{SH_{a_i}}(s,a) \times M_{SH_{a_i}}(s,a)}{\sum_{i=1}^{N} M_{SH_{a_i}}(s,a)} \tag{3}$$

In Eq.3 , N is the number of agents. The above formula shows that the Q-value for the pair s and a in the solo herding table is the weighted average of Q-value in all agents. The M shows the number of times pairs s and a occur for agent i in solo herding behavior where $1 \leqslant i \leqslant N$

5.2. *Effectively map*

In player agent, if the action is selected from Q-table, it needs to be mapped to an action appropriate to the current situation. As can be seen in Fig. 3, the action mapper unit is responsible for this.

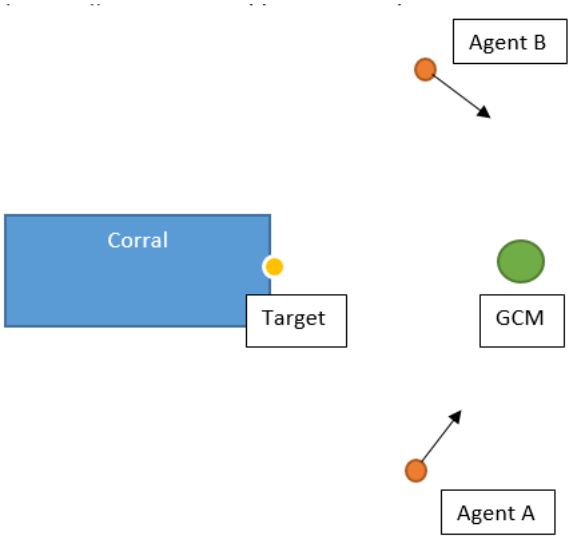

Fig. 9. A sample of "Effectively map"

As can be seen in Fig. 9, imagine a scenario in which agent A performs the northeast action and updates its solo following the Q-table. After aggregating the solo following Q-tables by coordinator agent and sending a single one

Table 1
Evaluated scenarios

Scenario1	130 cows and 160 obstacles
Scenario2	400 cows and 160 obstacles
Scenario3	400 cows and 600 obstacles
state_space1	d = 10 and a = 5
state_space2	d = 20 and a = 10

to all agents, on a time step when agent B is in the same state with agent A and wants to use agent A's experience, if it chooses the northeast action, the experience is not transferred correctly. The need for a mapping function that maps the action of a Q-table to an action appropriate to the current situation is now clearly felt. Since there is a model for predicting the movement of cows , the action can be considered as a difference between the current state and the next state. If the action was considered as the change of the current state and the predicted one, in the action mapping unit, the action was selected making the closest changes compared to current changes.

6. Heuristic techniques

When using reinforcement learning approaches, the use of heuristic algorithms will increase speed and improve efficiency. The heuristic algorithms used are rotational motion and middle action. The rotational motion algorithm consists of agent(s) that are in the behavior of solo or group following and instead of acting randomly, they choose an action that puts them in the right position behind the herd to guide the herd to its destination easily. The middle action algorithm consists of agent(s) that are in the behavior of solo or group herding and instead of acting randomly, they choose an action that its direction is closer to the direction of the line between GCM (global center of mass) and destination. The efficiency of the agent's knowledge transfer mechanism was evaluated by comparing success rates when using heuristic algorithms and knowledge transfer.

7. Experimental results and analysis

Each game will run 50,000 iterations. We tried to examine the effect of knowledge transfer and the way of calculating the state by simulating the game in different situations. The environmental conditions and the size of the state space were changed according to Table 1.

As can be seen in Fig. 10 to Fig. 13, knowledge transfer has a significant impact on the success rate of agents and convergence will occur faster as knowledge is transferred between agents. Criteria for evaluating knowledge transfer are:

- Jumpstart: the rate of improvement of basic efficiency when using knowledge transfer compared to when knowledge transfer was not used,
- Transfer ratio: the rate of total rewards collected when using knowledge transfer could be described mathematically as Eq.4:

$$R = \frac{S_{withtransfer} - S_{withouttransfer}}{S_{withouttransfer}} \tag{4}$$

In Fig. 10 to 11 the effect of d and a values on the success rate was investigated. When d = 10 and a = 5 the size of state space is equal to 7056 and when d = 20 and a = 10 the size of state space is equal to 882 and if the concept of state abstraction was not used, it will be equal to 1,800,000. As can be seen, in both cases, there is a significant reduction in the size of the state space.

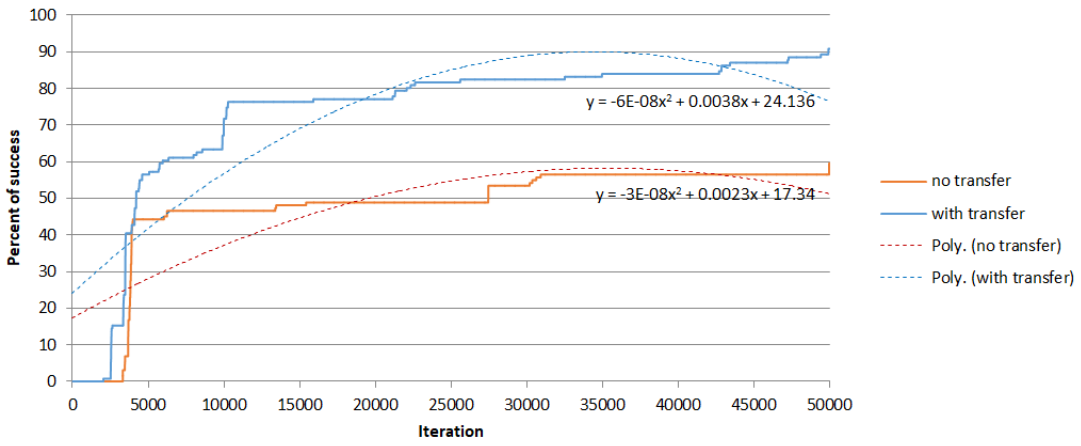

Fig. 10. Scenario1 with state_space1

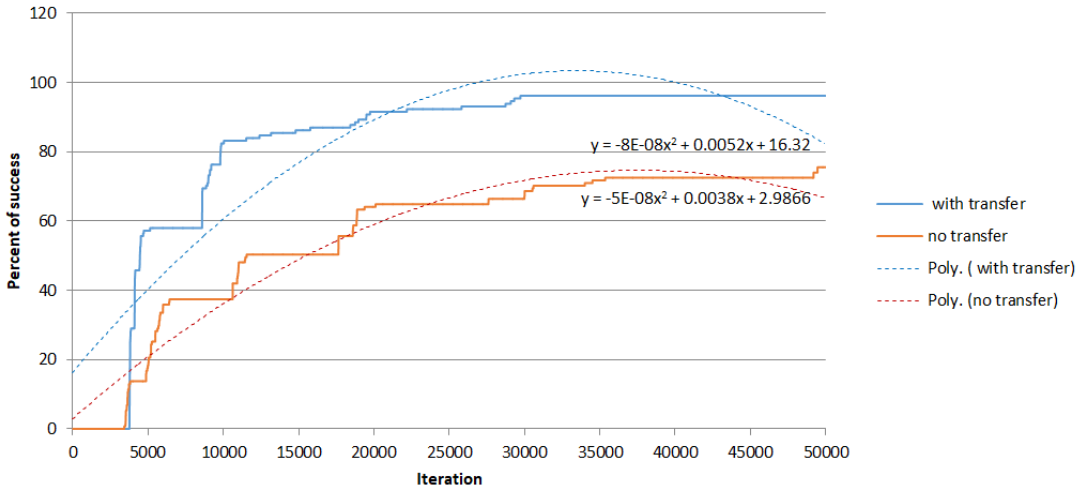

Fig. 11. Scenario1 with state_space2

When the values of $d = 20$ and $a = 10$ were used, better results were obtained because the number of similar cases increased and that enhanced the performance of the reinforcement learning algorithm. The values of $d = 20$ and $a = 10$ were used to examine the effect of knowledge transfer because increasing the number of similar cases has a greater effect on knowledge transfer. The impact of knowledge transfer is now being examined on the success of multi-agent system performance, according to the Fig. 11, when 130 cows are scattered in the environment and the number of obstacles is 160, the Jumpstart is about 29% when using knowledge transfer and 14% when not using knowledge transfer and the knowledge transfer rate is equal to 0.414. As can be seen in the Fig. 12, when 400 cows are scattered in the environment and the number of obstacles is 160, the Jumpstart is about 66% when using knowledge transfer and 24% when not using knowledge transfer and the knowledge transfer rate is equal to 0.598. As can be seen, knowledge transfer has increased the speed of convergence as well as increased efficiency. As can be observed in Fig. 14, that the effect of knowledge transfer is more significant when the number of cows scattered in the environment is greater, and the rate of knowledge transfer is greater when the number of cows in the environment is higher.

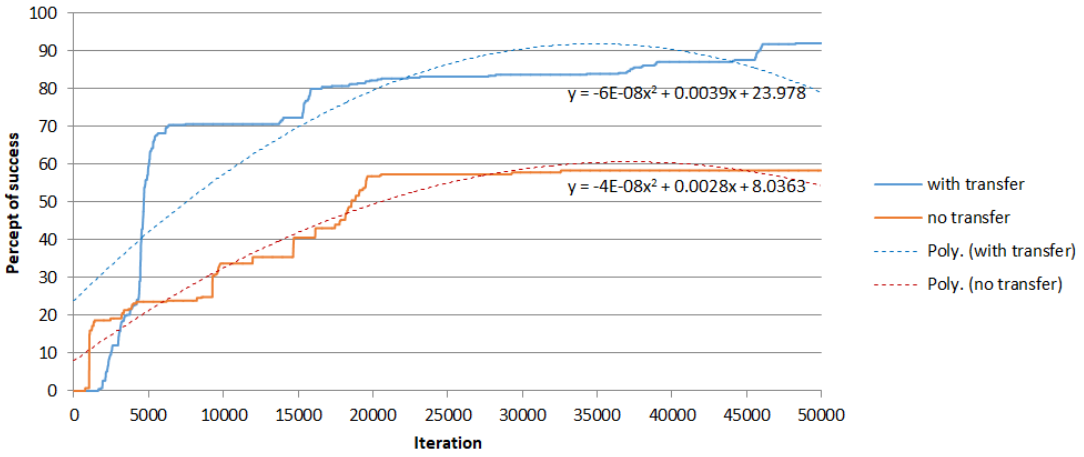

Fig. 12. Scenario2 with state_space2

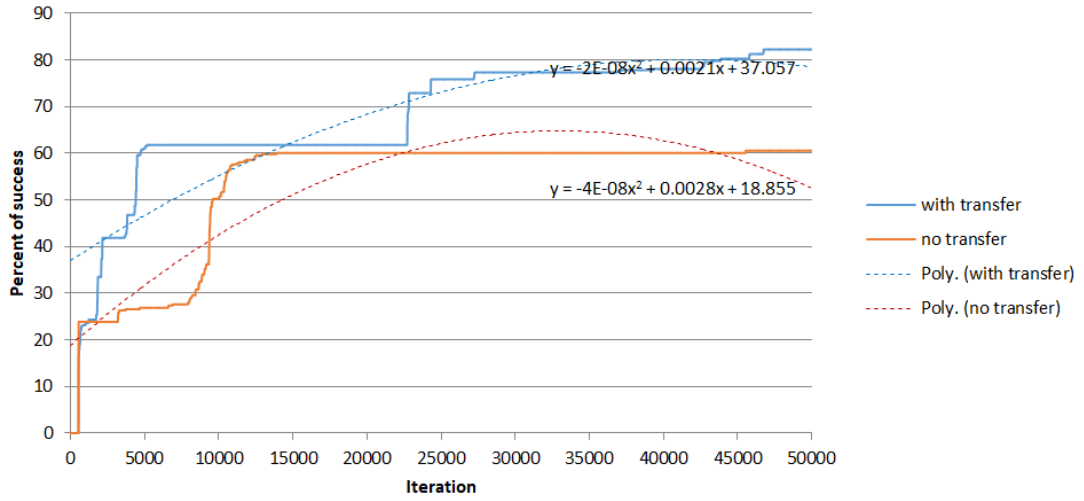

Fig. 13. Scenario3 with state_space2

In Fig. 13, the number of obstacles was almost quadrupled, when 400 cows are scattered in the environment and 600 obstacles are in their way, the Jumpstart is about 42% when using knowledge transfer and 24% when not using knowledge transfer and the knowledge transfer rate is equal to 0.312. As can be seen in the Fig. 15, when the number of obstacles is quadrupled, a good performance through knowledge transfer could be achieved, it is noticed that the functions defined for rewarding each behavior have a good performance.

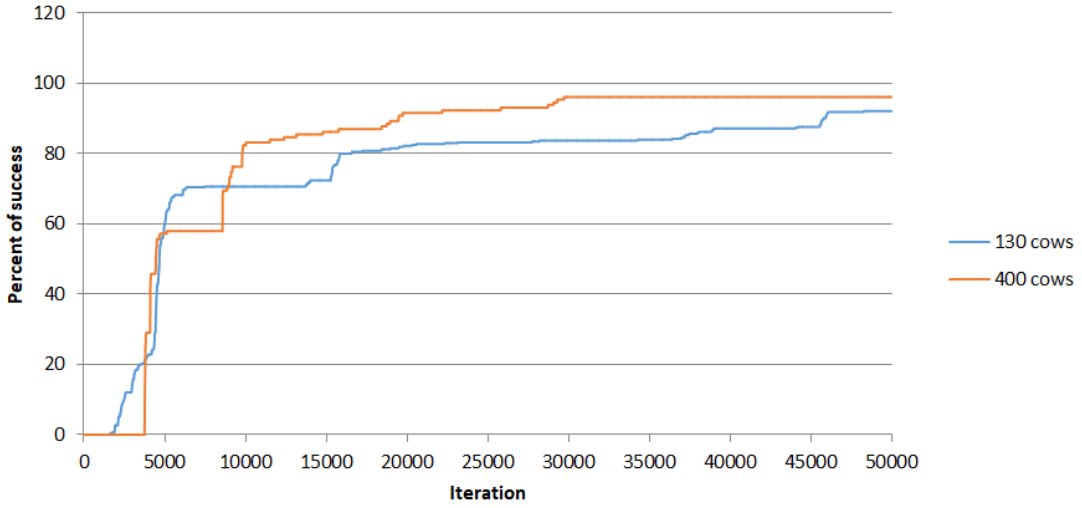

Fig. 14. Comparison between scenario1 and scenario2

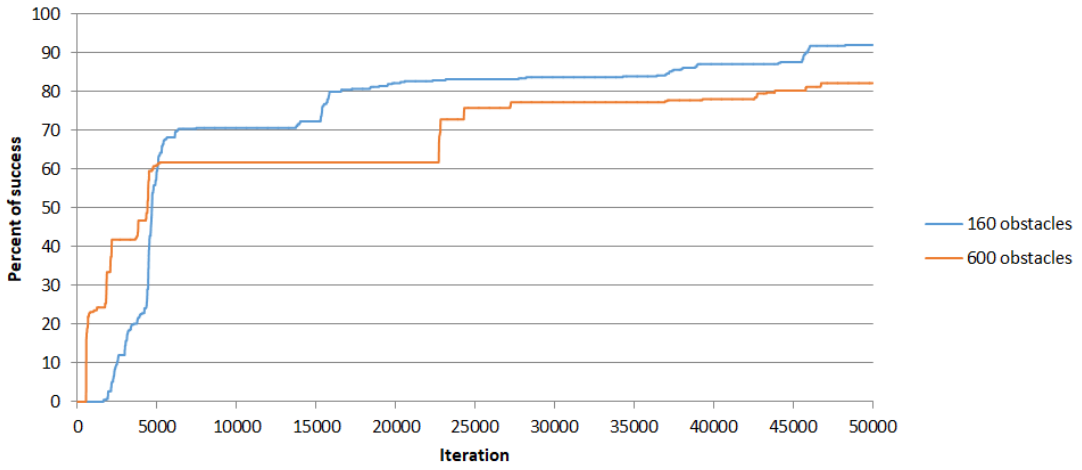

Fig. 15. Comparison between scenario2 and scenario3

According to Fig. 16, the performance of the knowledge transfer mechanism could be examined, while using the heuristic functions, the actions are selected correctly, and considering the proximity of the two charts, it can be concluded that the action mapping function works properly. In Table 2, the summary of Fig. 10 to Fig. 13 could be seen.

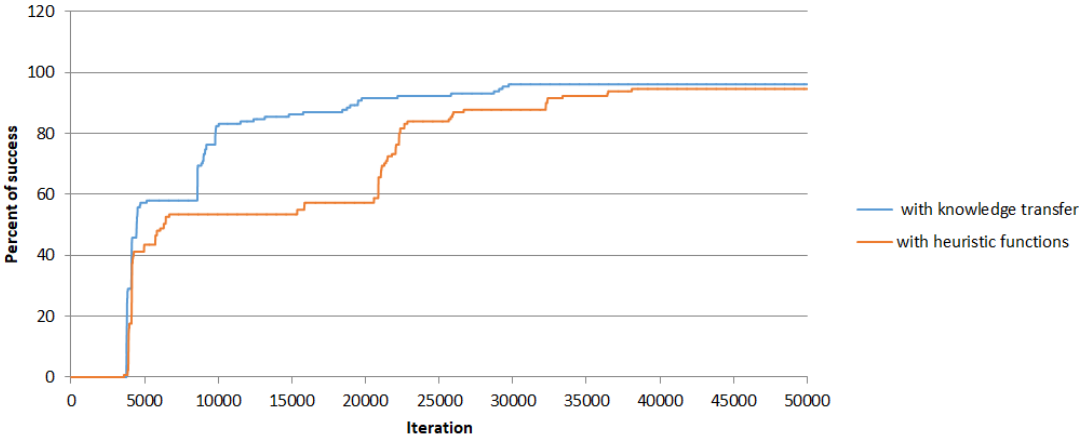

Fig. 16. Scenario1 with state_space2

Table 2
Scenario1 with state_space2

Condition	Scenario	State space	Success(%)	Convergence iteration
With transfer	One	One	90.83	49913
		Two	96.18	29726
	Two	Two	92.11	48278
	Three	Two	82.26	46764
No transfer	One	One	59.54	49958
		Two	75.57	49458
	Two	Two	58.37	32574
	Three	Two	60.59	45553

8. Time complexity

At the beginning of the implementation without considering the transfer of knowledge between agents, it was assumed that whenever an agent sees its allied agent in its line of sight, it will send a message to start cooperation. In this case, the time complexity is in each time step equal to $O(L, M^2))$, where L is line of sight and M is the number of agents in a group $1 \leqslant M \leqslant N$. We changed the initial implementation and prevented duplicate collaboration messages by adding a flag. In this case the time complexity is in each time step equal to $O(L, M)$. When the coordinator agent is added to the environment as an external agent and the knowledge transfer mechanism is used, at each time step all agents send their Q-tables to the coordinator agent and the coordinator agent also sends the aggregated Q-table to all agents so the time complexity is equal to $O(L, M + 2N)$ in each time step where L is line of sight and M is the number of agents in a group. Given that when we use knowledge transfer we can achieve the final success of about 92% (if there are 406 cows) and 96% (if there are 131 cows) so we can avoid increasing the complexity in order to increase efficiency.

9. Conclusion and future work

Solving complex problems, becoming a major challenge in the field of artificial intelligence. The shepherding problem is one such example due to its large state space as well as a dynamic and uncertain environment. We suggest that a solution to all the challenges mentioned above is the development of an appropriate communication framework between the agents of a multi-agent system. Our results demonstrate that we were able to significantly reduce the size of the state space by using the concept of state abstraction, and also through the calculation of the state of the agent or agents, we were able to increase the number of similar states, thereby increasing the efficiency of the reinforcement learning algorithm. We also showed that by transferring knowledge between agents we can reduce the cost of exploration and also increase the convergence speed and efficiency of our algorithm. For future work, in our implementation the agent's switching behavior was hard-coded in an if statement. As the number of behaviors increase, it makes sense to train agents to switch between different behaviors. Moreover, implementations could be done to reduce the messages exchanged between agents.

References

[1] J. Jin and X. Ma, A multi-objective agent-based control approach with application in intelligent traffic signal system, *IEEE Transactions on Intelligent Transportation Systems* (2019), 3900–3912.

[2] S. Qi and S. Zhu, Intent-aware multi-agent reinforcement learning., *IEEE International Conference on Robotics and Automation* (2018), 7533–7540.

[3] X. Liu, J. Yu, Z. Feng and Y. Gao,). Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing, *China Communications* (2020), 220–236.

[4] R. Sutton and A. Barto, *Introduction to reinforcement learning*, Cambridge: MIT press, Canada, 1998.

[5] W. Jiang, V. Narayanan and J. Li, Model Learning and Knowledge Sharing for Cooperative Multiagent Systems in Stochastic Environment, *IEEE Transactions on Cybernetics.* (2020).

[6] K. Shao, Y. Zhu and D. Zhao, Starcraft micromanagement with reinforcement learning and curriculum transfer learning, *IEEE Transactions on Emerging Topics in Computational Intelligence* (2018), 73–84.

[7] H. Suay and S. Chernova, Effect of human guidance and state space size on interactive reinforcement learning, *IEEE Ro-Man* (2011), 1–6.

[8] W. Jiang, V. Narayanan and J. Li, Model Learning and Knowledge Sharing for Cooperative Multiagent Systems in Stochastic Environment, *IEEE Transactions on Cybernetics* (2020).

[9] O. Bayazit, J. Lien and N. Amato, Roadmapbased flocking for complex environments, *10th Pacific Conference on Computer Graphics and Applications* (2002), 104–113–.

[10] T. Miki and T. Nakamura, An effective simple shepherding algorithm suitable for implementation to a multi-mmobile robot system, *First International Conference on Innovative Computing, Information and Control* **3** (2006), 161–165.

[11] N. Long, K. Sammut and D. Sgarioto, A Comprehensive Review of Shepherding as a Bio-Inspired Swarm-Robotics Guidance Approach, *IEEE Transactions on Emerging Topics in Computational Intelligence* (2020).

[12] S. Razali, Q. Meng and S. Yang, A refined immune systems inspired model for multi-robot shepherding, *Second World Congress on Nature and Biologically Inspired Computing* (2010), 473–478.

[13] S. Razali, Q. Meng and S. Yang, Immune-inspired cooperative mechanism with refined low-level behaviors for multi-robot shepherding, *International Journal of Computational Intelligence and Applications* (2012).

[14] D. Strömbom, R. Mann and A. Wilson, Solving the shepherding problem: heuristics for herding autonomous, interacting agents, *Journal of the royal society interface* (2014).

[15] A. Pierson and M. Schwager, Bio-inspired non-cooperative multi-robot herding, *ICRA* (2015), 1843–1849.

[16] K. Fujioka and S. Hayashi, Effective shepherding behaviours using multi-agent systems, *2016 IEEE Region 10 Conference (TENCON)* (2016), 3179–3182.

[17] W. Lee and D. Kim, Autonomous shepherding behaviors of multiple target steering robots, *Sensors* (2017).

[18] H. Hoshi, I. Iimura, S. Nakayama and Y. Moriyama, Robustness of Herding Algorithm with a Single Shepherd Regarding Agents' Moving Speeds, *Journal of Signal Processing* (2018), 327–335.

[19] H. Hoshi, I. Iimura and S. Nakayama, Computer Simulation Based Robustness Comparison Regarding Agents' Moving-Speeds in Two-and Three-Dimensional Herding Algorithms, *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)* (2018), 1307–1314.