Abstract MDP Reward Shaping for Multi-Agent Reinforcement Learning

Kyriakos Efthymiadis, Sam Devlin and Daniel Kudenko

Department of Computer Science, The University of York, UK

Abstract. Reward shaping has been shown to significantly improve an agent's performance in reinforcement learning. As attention is shifting from tabula-rasa approaches to methods where some heuristic domain knowledge can be given to agents, an important problem that arises is how to generate a useful potential function. Previous research demonstrated the use of plan-based reward shaping in multi-agent reinforcement learning (MARL), where STRIPS planning was used to generate a potential function. The results showed that potential functions based on joint plans can improve an agent's performance. When using individual plans however, the agents face conflicting goals which can have a detrimental effect in performance. In this paper we present the use of abstract MDPs as a method to provide heuristic knowledge in MARL and how it can be utilised for conflict resolution when agent communication and goal-shaping is not possible.

Keywords: multi-agent reinforcement learning, reward shaping, abstract MDP

1 Introduction

Reinforcement learning has proven to be a successful technique when an agent needs to act and improve in a given environment. The agent receives feedback about its behaviour in terms of rewards through constant interaction with the environment. Traditional reinforcement learning assumes the agent has no prior knowledge about the environment it is acting on. Nevertheless, in many cases (potentially abstract and heuristic) domain knowledge of the reinforcement learning tasks is available, and can be used to improve the learning performance.

In earlier work on *knowledge-based reinforcement learning* [1–3] it was demonstrated that the incorporation of domain knowledge in reinforcement learning via reward shaping can significantly improve the speed of converging to an optimal policy. Reward shaping is the process of providing prior knowledge to an agent through additional rewards. These rewards help direct an agent's exploration, minimising the number of sub-optimal steps it takes and so directing it towards the optimal policy quicker. However, the problem remains of how to design a useful potential function.

Previous research demonstrated the use of plan-based reward shaping for multi-agent reinforcement learning [4]. This method uses STRIPS planning to

generate a potential function in order to shape the agents. The generation of multi-agent plans can occur within one centralised agent or spread amongst a number of agents [5, 6]. The centralised approach benefits from full observation making it able to, where possible, satisfy all agents' goals without conflict. However, this approach requires sharing of information, such as goals and abilities, that agents in a multi-agent system often will not want to share.

The alternative approach, allowing each agent to make their own plans, will tend to generate conflicting plans. This can have an impact in behavior with the agents not being able to coordinate efficiently [4] and fail to reach a good enough policy similar to that of receiving joint plans. Many methods of coordination have been attempted including, amongst others, social laws [7], negotiation [6] and contingency planning [8] but still this remains an ongoing area of active research.

We are interested in those settings where information sharing is not allowed and the agents are agnostic to other learning entities acting in the environment.

In this paper we propose the use of a modified version of abstract MDPs for reward shaping in MARL. Marthi [9] proposed the use of abstract MDPs as a means to automatically provide guidance to an agent, by solving an abstract MDP of the environment and use the resulting value function to shape the agent. We demonstrate empirically that the agents using abstract MDP reward shaping despite receiving decentralised shaping which contains conflicting goals, manage to efficiently overcome them and coordinate to learn a much better policy compared to the agents using plan-based reward shaping, which fail to do so. We show that abstract MDPs as a reward shaping function can be used in decentralised planning for decentralised agents as a mean of coordination.

2 Background

2.1 Reinforcement Learning

Reinforcement learning is a method where an agent learns by receiving rewards or punishments through continuous interaction with the environment [10]. The agent receives a numeric feedback relative to its actions and in time learns how to optimise its action choices. Typically reinforcement learning uses a Markov Decision Process (MDP) as a mathematical model [11].

An MDP is a tuple $\langle S, A, T, R \rangle$, where S is the state space, A is the action space, $T(s, a, s') = Pr(s'|s, a)$ is the probability that action a in state s will lead to state s', and $R(s, a, s')$ is the immediate reward r received when action a taken in state s results in a transition to state s'. The problem of solving an MDP is to find a policy (i.e., mapping from states to actions) which maximises the accumulated reward. When the environment dynamics (transition probabilities and reward function) are available, this task can be solved using dynamic programming [12].

When the environment dynamics are not available, as with most real problem domains, dynamic programming cannot be used. However, the concept of an iterative approach remains the backbone of the majority of reinforcement learning algorithms. These algorithms apply so called temporal-difference updates to

propagate information about values of states, $V(s)$, or state-action pairs, $Q(s, a)$. These updates are based on the difference of the two temporally different estimates of a particular state or state-action value. The SARSA algorithm is such a method [10]. After each real transition, $(s, a) \rightarrow (s', r)$, in the environment, it updates state-action values by the formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)] \tag{1}$$

where α is the learning rate and γ is the discount factor. It modifies the value of taking action a in state s, when after executing this action the environment returned reward r, moved to a new state s', and action a' was chosen in state s'.

It is important whilst learning in an environment to balance exploration of new state-action pairs with exploitation of those which are already known to receive high rewards. A common method of doing so is $\epsilon-$greedy exploration. When using this method the agent explores, with probability ϵ, by choosing a random action or exploits its current knowledge, with probability $1 - \epsilon$, by choosing the highest value action for the current state [10].

Temporal-difference algorithms, such as SARSA, only update the single latest state-action pair. In environments where rewards are sparse, many episodes may be required for the true value of a policy to propagate sufficiently. To speed up this process, a method known as eligibility traces keeps a record of previous state-action pairs that have occurred and are therefore eligible for update when a reward is received. The eligibility of the latest state-action pair is set to 1 and all other state-action pairs' eligibility is multiplied by λ (where $\lambda \leq 1$). When an action is completed all state-action pairs are updated by the temporal difference multiplied by their eligibility and so Q-values propagate quicker [10].

Typically, reinforcement learning agents are deployed with no prior knowledge. The assumption is that the developer has no knowledge of how the agent(s) should behave. However, more often than not, this is not the case. We are interested in *knowledge-based reinforcement learning*, an area where this assumption is removed and informed agents can benefit from prior knowledge.

2.2 Reward Shaping

One common method of imparting knowledge to a reinforcement learning agent is reward shaping. In this approach, an additional reward representative of prior knowledge is given to the agent to reduce the number of suboptimal actions made and so reduce the time needed to learn [13, 14]. This concept can be represented by the following formula for the SARSA algorithm:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + F(s, s') + \gamma Q(s', a') - Q(s, a)] \tag{2}$$

where $F(s, s')$ is the general form of any state-based shaping reward.

Even though reward shaping has been powerful in many experiments it quickly became apparent that, when used improperly, it can change the optimal policy [14]. To deal with such problems, potential-based reward shaping was

proposed [13] as the difference of some potential function Φ defined over a source s and a destination state s':

$$F(s, s') = \gamma\Phi(s') - \Phi(s) \tag{3}$$

where γ must be the same discount factor as used in the agent's update rule (see Equation 1).

This formulation of reward shaping has been proven to not alter the optimal policy of a single agent in both infinite- and finite- state MDPs [13].

Wiewiora [15] later proved that an agent learning with potential-based reward shaping and no knowledge-based Q-table initialisation will behave identically to an agent without reward shaping when the latter agent's value function is initialised with the same potential function.

More recent work on potential-based reward shaping, has removed the assumptions of a single agent acting alone and of a static potential function from the original proof. In multiagent systems, it has been proven that potential-based reward shaping can change the joint policy learnt but does not change the Nash equilibria of the underlying game [16].

With a dynamic potential function [17], it has been proven that the existing single and multi agent guarantees are maintained provided the potential of a state is evaluated at the time the state is entered and used in both the potential calculation on entering and exiting the state. Furthermore, potential-based reward shaping with a dynamic potential function is not equivalent to Q-table initialisation.

Reward shaping is typically implemented bespoke for each new environment using domain-specific heuristic knowledge [1, 14] but some attempts have been made to automate [18, 9] and semi-automate [3] the encoding of knowledge into a reward signal. Automating the process requires no previous knowledge and can be applied generally to any problem domain. The results are typically better than without shaping but less than agents shaped by prior knowledge. Semi-automated methods require prior knowledge to be put in but then automate the transformation of this knowledge into a potential function.

2.3 Plan-Based Reward Shaping

Plan-based reward shaping, an established semi-automated method, generates a potential function from prior knowledge represented as a high level STRIPS plan. STRIPS is a highly popular and widely used and studied formalism used to express automated planning instances. We briefly explain the STRIPS formalism and refer the reader to [19] for further details, as well as a large collection of books, papers etc. on this topic.

The description of the planning problem in STRIPS includes the operators, actions, which specify the behaviour of the system, and start and goal states [19]. Actions have a set of preconditions which have to be satisfied for the action to be executable, and a set of effects which are made true or false by executing the action. The start state contains a set of conditions which are initially true,

and the goal state consists of conditions which have to be true or false for the state to be classified as a goal state.

The output of the planner is the sequence of actions, that will satisfy the conditions set at the goal state. The STRIPS plan is translated from an action-based, to a state-based representation so that, whilst acting, an agent's current state can be mapped to a step in the plan[1], as illustrated in Figure 1.

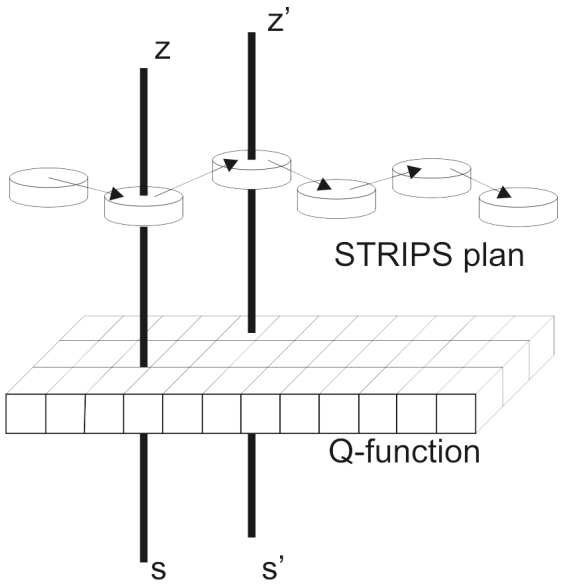

Fig. 1. Plan-Based Reward Shaping.

The potential of the agent's current state then becomes:

$$\Phi(s) = CurrentStepInPlan * \omega \tag{4}$$

where $CurrentStepInPlan$ is the corresponding state in the state-based representation of the agent's plan and ω is a scaling factor.

To preserve the theoretical guarantees of potential-based reward shaping, the potential of all terminal states is set to zero.

These potentials are then used as in Equation 3 to calculate the additional reward given to the agent and so encourage it to follow the plan without altering the agent's original goal. The process of learning the low-level actions necessary to execute a high-level plan is significantly easier than learning the low-level actions to maximise reward in an unknown environment and so with this knowledge agents tend to learn the optimal policy quicker. Furthermore,

[1] Please note that one step in the plan will map to many low level states. Therefore, the agent must learn how to execute this plan at the low level.

as many developers are already familiar with STRIPS planners, the process of implementing potential-based reward shaping is now more accessible and less domain specific [3].

2.4 Abstract MDP Reward Shaping

Marthi [9] proposed a general automatic framework to learn the potential function by solving an abstract MDP. The shaping algorithm obtains the potential function by firstly sampling the environment in order to learn dynamics for options (i.e. actions at the abstract level) and secondly solving an abstract MDP. Options can be defined as policies of low level actions. Once the agent spends a number of episodes sampling the environment, it uses the resulting value function as a source for reward shaping.

We are interested in cases where domain knowledge comes from domain experts and therefore we have modified the abstract MDP reward shaping method to incorporate prior knowledge. We assume options to be primitive deterministic actions at an abstract level and therefore computation of their dynamics can be omitted.

In addition, by providing an abstraction of the low-level states of the environment to high-level abstract states[2], the abstract MDP can be solved using dynamic programming before the main learning process begins and the obtained value function is used directly as the potential function:

$$\Phi(s) = V(z) * \omega, \tag{5}$$

where $V(z)$ is the value function over the abstract state space Z and it represents a solution to the corresponding MDP-based planning problem, and ω is an optional scaling factor. Modifying the method in this way, results in the agent not having to spend time randomly exploring the environment to estimate its dynamics and receives guidance the moment it starts acting. The abstract MDP task can be solved using the following formula which is a special case of value iteration:

$$V_{k+1}(z) = \max_{z'}[R_{zz'} + \gamma V_k(z')], \tag{6}$$

with $R_{zz'}$ the reward received when transitioning to z' from z, γ the discount factor and $V_k(z)$ the value of state z at time k.

These potentials are then used as in Equation 3 to calculate the additional reward to be given to the agent. Learning low-level actions in order to satisfy a high-level abstract MDP is significantly easier than learning low-level actions to maximise rewards in an unknown environment and as a result agents tend to learn a policy quicker.

[2] Please note that a high-level abstract state will map to many low level states. Therefore, even when provided with the correct knowledge, the agent still needs to learn the optimal path to finish the episode.

3 Experimental Design

Evaluation Domain

We evaluate the reward shaping algorithms on an extended version of the navigation maze problem, the flag-collection domain. The agents are modelled at a starting position from where they must move to the goal position. In between, the agents needs to collect flags which are spread throughout the maze. During an episode, at each time step, the agents are given their current location and the flags they have already collected. From this they must decide to move up, down, left or right and will deterministically complete their move provided they do not collide with a wall. Regardless of the number of flags collected, the scenario ends when both agents reach the goal position. At this time the agents receives a reward equal to one hundred times the number of flags which were collected.

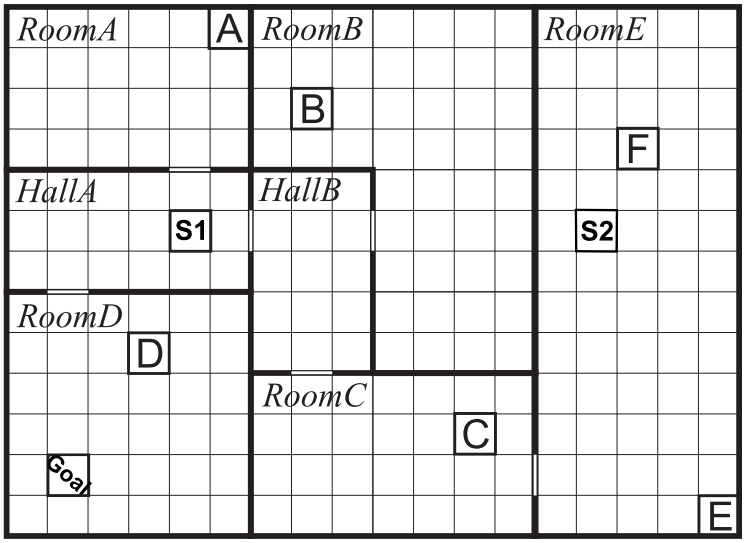

Fig. 2. Flag-Collection Domain.

At a more abstract level, the maze can be thought of as a house with doors between rooms. Each room might, or might not contain a flag and it is up to the agents to find where the flags are located.

```
0        robot_in(hallA)
1        robot_in(roomD)
2        robot_in(roomD)  taken(flagD)
```

Listing 1.1. Example Partial State-Based Plan

Figure 2 shows the layout of a simple version of the domain in which rooms are labelled *RoomA-E* and *HallA-B*, flags are labelled *A-F*, *S1* and *S2* are the starting positions of the agents and *G* is the goal position.

```
robot_in(hallA)                    96
robot_in(roomD)                    98
robot_in(roomD)  taken(flagD)      100
```

Listing 1.2. Example Partial Value Function

Given this domain, the room the agent is in and the flags it has picked up is used as a state abstraction to be used in Equation 5. The actions abstraction is the transitions the agent is allowed to preform while navigating across different rooms in the maze i.e. they are moves from one room to another. These are used in order to solve the high-level abstract MDP with the resulting value function being used to shape the agent.

A partial example of the expected value function to be used for shaping is given in Listing 1.2 with $V(z)$ used in Equation 5 shown in the right hand column and a partial STRIPS plan is shown in 1.1 with the left hand side showing the step in the plan to be used in Equation 4.

Assumptions

A direct translation of the low level states in the grid to the abstract high level states is assumed. For instance, in this domain the high level knowledge includes rooms, connections between rooms within the maze and the rooms which flags should be present in. Whilst the translation of low level to high level states allows an agent to lookup which room or hall it is in from the exact location given in its state representation.

The domain is considered to be static i.e. there are no external events not controlled by the agent which can at any point change the environment.

We do not assume full observability or knowledge of the transition and reward functions. In addition, we do not assume deterministic transitions.

The domain we have chosen allows the agent's behaviour to be efficiently extracted and analysed, thus providing useful insight especially when dealing with novel approaches. Therefore, the method presented in this paper is suitable for any real-world problem matching these properties. Such a domain can also be seen as a proxy to the real world [20]. Following this approach to empirical evaluations is commonly adopted in the related literature e.g. [21].

3.1 Results

A series of experiments were conducted in order to assess the performance of the abstract MDP reward shaping method for MARL. Our agent is compared

against an agent using plan-based reward shaping. The comparison is based on the performance in terms of discounted goal reward[3].

In order to be fair when comparing the two approaches, the same state abstraction function and options/actions are used both in the plan-based and abstract MDP methods.

In all our experiments, we have set the scaling factor of the abstract MDP method shown in Equation 5 to:

$$\omega = MaxReward/NumStatesInMDP \tag{7}$$

and the scaling factor of the plan-based method shown in Equation 4 to:

$$\omega = MaxReward/NumStepsInPlan \tag{8}$$

As the scaling factor affects how likely the agent is to follow the heuristic knowledge, maintaining a constant maximum across all heuristics compared ensures a fair comparison. For environments with an unknown maximum reward the scaling factor ω can be set experimentally or based on the designer's confidence in the heuristic.

All agents implemented SARSA with $\epsilon-$greedy action selection and eligibility traces [10]. For all experiments, the agents' parameters were set such that $\alpha = 0.1$, $\gamma = 0.99$, $\epsilon = 0.3$ and $\lambda = 0.4$. The experiments were run for 10 iterations each lasting $50,000$ episodes.

These methods, however, do not require the use of SARSA, $\epsilon-$greedy action selection or eligibility traces. Potential-based reward shaping has previously been proven with Q-learning, RMax and any action selection method that chooses actions based on relative difference and not absolute magnitude [22]. Furthermore, it has been shown to work before without eligibility traces [1, 13]. For clarity all the graphs only display results up to 20000 episodes, after this time no significant change in behaviour occurred in any of the experiments[4]. The graphs also include error bars showing the standard error of the mean.

We have tested the agents using the flag collection domain described earlier and shown in Figure 2. In addition we use two scaled up version of this maze; a maze with 12 flags and 7 rooms and a maze with 12 flags and 12 rooms. We are interested in those settings where information sharing is not allowed and the agents are agnostic to other learning entities in the environment. To set an upper bound on performance however, we also include a setting in which agents receive plan-based reward shaping from a joint-plan generated by a centralised agent. Figures 3, 4 and 5 show the performance of the agents.

[3] Please note the agents' illustrated performance does not reach the maximum reward given as the value presented is discounted by the time it takes the agents to complete the episode.

[4] The reader should note that if the agents were let to run an infinite amount of episodes, they eventually converge to the same policy as the reinforcement learning theory suggests.

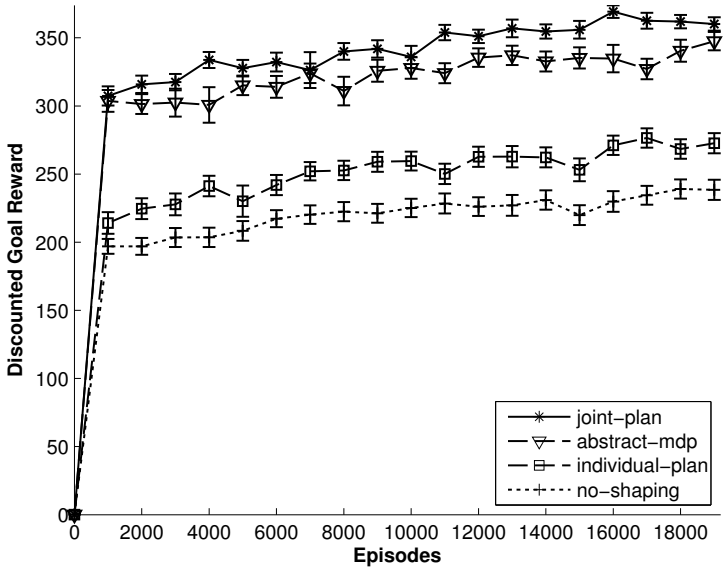

Fig. 3. Multi-agent flag-collection domain with 6 flags and 7 rooms.

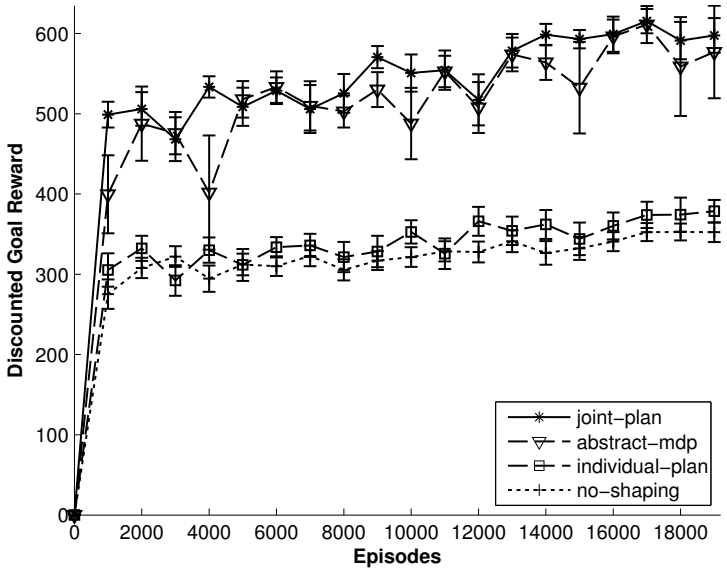

Fig. 4. Multi-agent flag-collection domain with 12 flags and 7 rooms.

3.2 Discussion

The results show that the plan-based agent receiving individual shaping fails to reach a satisfying performance. Careful examination shows that the agents fail

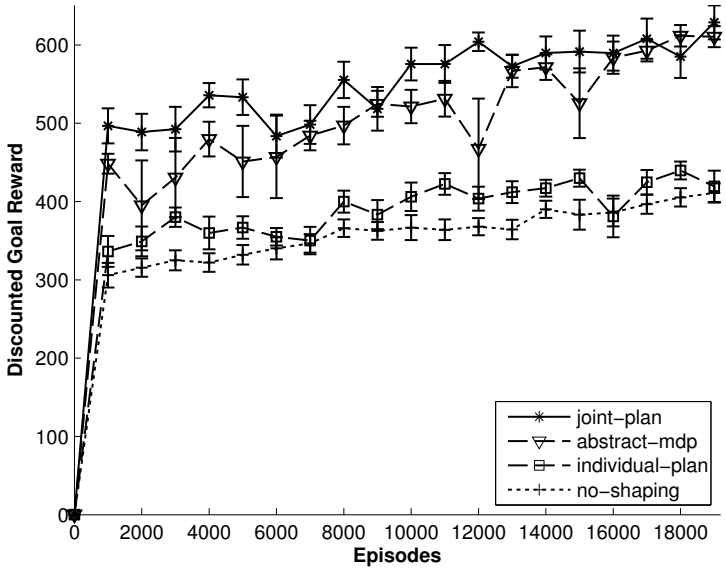

Fig. 5. Multi-agent flag-collection domain with 12 flags and 12 rooms.

to coordinate and one of them opts out and heads to the goal location, while the other agent collects all the flags. This is due to the way plan-based shaping provides extra rewards. Certain goals in the plan cannot be satisfied and as a result only one agent is able to collect all the extra rewards and learn a better policy.

More specifically, when both agents receive individual plans, they are guided to collect all the flags that are present in the maze. The knowledge provided as a STRIPS plan contains a single path to the goal, i.e. a succession of high level states the agent will have to go through, similar to that shown in Listing1.1. As a result, if one of the steps cannot be satisfied by the agent, no other extra rewards can be given after that point. Consider the case where the second agent has picked up flag C. Any steps in the first agent's plan which contain flag C now cannot be satisfied and the agent is left without any reward shaping after that point.

The agents receiving abstract MDP shaping manage to achieve a performance similar to the agent receiving centralised shaping. Since the agent is agnostic to other learning entities in the environment, it follows that certain paths in the shaping function will not be encountered in simulation. For instance, consider that the second agent picks up flags E and F in the maze shown in Figure 2, this will have as a consequence the first agent never encountering those states where it has collected those flags. This does not have any impact in the agent's performance since there still exist other paths in the shaping function which lead to the goal position. This is due to the fact that, contrary to the plan-

based agent, the abstract MDP agent does not receive only a single path to the goal since the value function contains all the possible states the agent can be in along with their values. Therefore, both agents are able to learn a much better policy than the individual learners using plan-based shaping and the agents can cooperate efficiently without the need to use centralised shaping.

4 Conclusion

We have proposed the use of abstract MDP reward shaping for MARL. We have compared this approach to plan-based reward shaping and we are interested in the cases where sharing of information is not permitted.

We have demonstrated that the abstract MDP agents can learn to cooperate efficiently and eliminate conflicting goals while the plan-based method cannot reach similar performance. This difference in performance is attributed to the type of knowledge provided by the two methods. While both are considered decentralised methods for reward shaping, abstract MDPs provide multiple paths to the goal when the plan-based method provides only a single path and has a direct impact in performance due to the conflicting goals.

Therefore, abstract MDP reward shaping can be used not only as a method to impart domain knowledge in MARL, but also as a means of conflict resolution and cooperation in decentralised reward shaping.

References

1. Devlin, S., Grześ, M., Kudenko, D.: An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. Advances in Complex Systems (2011)
2. Efthymiadis, K., Kudenko, D.: Using plan-based reward shaping to learn strategies in starcraft: Broodwar. In: Computational Intelligence and Games (CIG), 2013 IEEE Conference on, IEEE (2013)
3. Grześ, M., Kudenko, D.: Plan-based reward shaping for reinforcement learning. In: Proceedings of the 4th IEEE International Conference on Intelligent Systems (IS'08), IEEE (2008) 22–29
4. Devlin, S., Kudenko, D.: Plan-based reward shaping for multi-agent reinforcement learning. In: In Proceedings of the AAMAS Workshop on Adaptive and Learning Agents (ALA). (2012)
5. Rosenschein, J.: Synchronization of multi-agent plans. In: Proceedings of the National Conference on Artificial Intelligence. (1982) 115–119
6. Ziparo, V.: Multi-Agent Planning. Technical report, University of Rome (2005)
7. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artificial Intelligence **73**(1-2) (1995) 231–252
8. Peot, M., Smith, D.: Conditional nonlinear planning. In: Artificial Intelligence Planning Systems: Proceedings of the First International Conference, Morgan Kaufmann Pub (1992) 189
9. Marthi, B.: Automatic shaping and decomposition of reward functions. In: Proceedings of the 24th International Conference on Machine learning, ACM (2007) 608

10. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
11. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley and Sons, Inc., New York, NY, USA (1994)
12. Bertsekas, D.P.: Dynamic Programming and Optimal Control (2 Vol Set). Athena Scientific, 3rd edition (2007)
13. Ng, A.Y., Harada, D., Russell, S.J.: Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of the 16th International Conference on Machine Learning. (1999) 278–287
14. Randløv, J., Alstrom, P.: Learning to drive a bicycle using reinforcement learning and shaping. In: Proceedings of the 15th International Conference on Machine Learning. (1998) 463–471
15. Wiewiora, E.: Potential-based shaping and Q-value initialization are equivalent. Journal of Artificial Intelligence Research **19**(1) (2003) 205–208
16. Devlin, S., Kudenko, D.: Theoretical considerations of potential-based reward shaping for multi-agent systems. In: Proceedings of The Tenth Annual International Conference on Autonomous Agents and Multiagent Systems. (2011)
17. Devlin, S., Kudenko, D.: Dynamic potential-based reward shaping. In: Proceedings of The Eleventh Annual International Conference on Autonomous Agents and Multiagent Systems. (2012)
18. Grześ, M., Kudenko, D.: Multigrid Reinforcement Learning with Reward Shaping. Artificial Neural Networks-ICANN 2008 (2008) 357–366
19. Fikes, R.E., Nilsson, N.J.: Strips: A new approach to the application of theorem proving to problem solving. Artificial intelligence **2**(3) (1972) 189–208
20. Pasula, H.M., Zettlemoyer, L.S., Kaelbling, L.P.: Learning symbolic models of stochastic domains. J. Artif. Intell. Res.(JAIR) **29** (2007) 309–352
21. Ryan, M.R.: Using abstract models of behaviours to automatically generate reinforcement learning hierarchies. In: ICML. Volume 2., Citeseer (2002) 522–529
22. Asmuth, J., Littman, M., Zinkov, R.: Potential-based shaping in model-based reinforcement learning. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence. (2008) 604–609