# Plan-Based Reward Shaping for Multi-Agent Reinforcement Learning

Jérôme Bastogne[1], Maxime Desclefs[1] and Simon Picard[1]

[1]Université Libre de Bruxelles, Boulevard du Triomphe - CP 212, 1050 Brussels, Belgium
jbastogn@ulb.ac.be, mdesclef@ulb.ac.be, spicard@ulb.ac.be

**Abstract**

Reward shaping is known to significantly improve an agent's performance in multi-agent reinforcement learning (MARL). This paper shows the benefits of using plan-based reward shaping in which a STRIPS planning knowledge is used. The results of agents using plan-based reward shaping will then be compared to a simple Reinforcement Learning (RL) agent with no domain knowledge and they will show how they outperform previous results.

## Introduction

Reinforcement learning agents don't get future feedback about how good was their decision taken in a precise state. This leads to a temporal problem as the agents will not know immediately which part of their decisions where the good ones (Grzes and Kudenko, 2011). RL, while being simple, presents some issues. The time taken by the agents to learn the right policy grows exponentially while adding new variables to the environment. When the state space is too vast, memory becomes an issue as well as the matrix for each state-action pair becomes too big and too many states need to be updated too frequently slowing down drastically the process.

A way of improving the converging speed is to provide prior knowledge to the agents through a method called reward shaping. Reward shaping is the addition of domain knowledge to reinforcement learning in a way that will minimize non-optimal behaviours and will fastly converge to the optimal one. In this article we will see how to effectively incorporate reward shaping in MARL using individual and joint plan based plans and we will combine them with a flag based heuristic(Devlin and Kudenko, 2004).

We will show that providing prior knowledge, while not always being possible due to heuristic problems, significantly increases agent's performances and speed of convergence.

## Study Case

We choose two agents who starts on the S1 and S2 squares respectively. Their objective is to reach the goal while trying to collect a maximum of flags while doing so. At each time step, they will move one square up, down, left or right unless they collide with a wall or the other agent. Each of our agents has to reach the goal to end an episode. When an agent reaches the goal, his episode is over and he will wait the other agent to finish his run. When they both reach the goal, they receive a reward equal to $100 * NumFlagsCollected$.

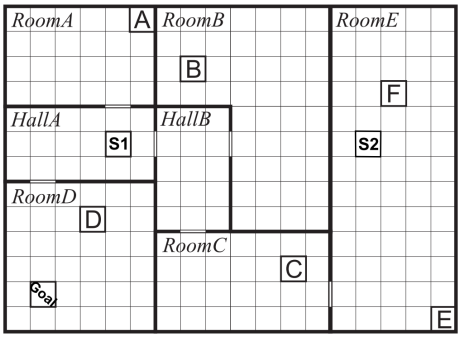


Figure 1: Multi-Agent, Flag-Collecting Problem Domain.

## Materials and Methods

### Reinforcement Learning

Reinforcement learning is a method of rewarding an agent for taking a good decision in a particular state. The better state the decision leads to, the better that decision will be rewarded. Therefore RL can be considered as a Markov Decision Process. Each agent remembers what rewards he got for each action in each state.

The possible actions here are going up, down, left and right. A state here, is a position (x, y) in the world space combined the current step in the plan. To each of these Q(s,a) state-action pairs is associated a Q-value translating the goodness of choosing that action while being in that state. The reinforcement algorithm is iterative and dynamic, therefore agents have to run multiple episodes where they have to reach the goal and where they are rewarded depending on the number of flags they got together. At

each step, we need to update Q(s,a) for future learning. A way of achieving this is by applying a temporal-difference update to Q(s,a) so that Q(s,a) will increase only if it leads to a better state than the actual one. This ensures that agents are willing of doing better while reaching the goal. There is an algorithm called SARSA that does this by updating Q-values as follows :

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

where $\alpha$ is the rate of learning and $\gamma$ is the discount factor. r is the reward returned by the environment. In our case, rewards will only be given when agents reach the final goal.

At the beginning, agents know nothing about their environment, therefore they will randomly travel through the world space. After few iterations some of their actions will have positive rewarded scores but might not be the most optimals. Therefore, a good balance between exploration and exploitation of the best rewarded actions must be found. The most common method is by using the $\epsilon$-greedy algorithm. It ensures that at each step, the best value action will be chosen with a probability 1- $\epsilon$ and otherways, the agent will explore a new action.

A well-known method to fasten this process is eligibility traces. When a reward is received, the lasts state-actions pairs that the agent has been walking through are also updated according to their temporal difference with the last state-action pair. Each Q-value is updated as follows :

$$Q(s,a) \leftarrow Q(s,a) + \alpha * \sigma(\gamma * \lambda)^t$$

where $\lambda$ has to be lower or equal to 1 (Singh and Sutton, 1996). Where $\sigma = r + \gamma Q(s',a') - Q(s,a)$ We choosed $\lambda = 0.4$ for our experiment.

## Reward Shaping

Reward shaping is a method in reinforcement learning that attributes additionnal rewards to the agents that should guide them to find their optimality and in a much faster way. The addition of reward shaping to reinforcement learning changes the SARSA algorithm formula as follows:

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

where $F(s,s')$ is a difference of potentials defined as follows :

$$F(s,s') = \gamma\phi(s') - \phi(s)$$

where $\phi$ is some function over states. The difference of potentials is usefull for avoiding cycles (Andrew Y. Ng and Russell, 1999).

## Plan-Based Reward Shaping

For plan-based reward shaping, agents are given STRIPS plans that transposes into a list of states as shown on Figure 2.

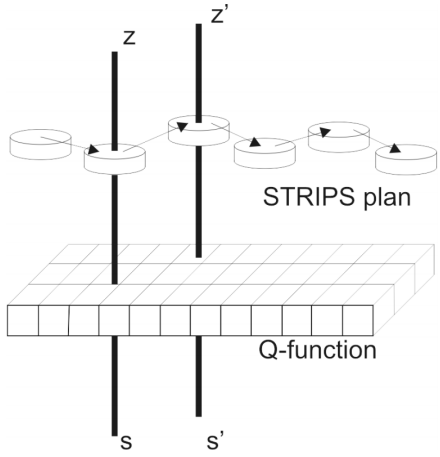


Figure 2: Plan-base reward shaping.

It consists in a set of subgoals that the agent should be focusing on in the specified order to increase the total ending reward. In our experiment, subgoals are : going from a room to another, and collecting a flag in the current room when specified. Examples of individual and joint plans we will be using are given on figures 3 and 4.

Listing 3: Individual Plan for Agent 1 Starting in HallA

```
MOVE( hallA , hallB )
MOVE( hallB , roomC )
TAKE( flagC , roomC )
MOVE( roomC , roomE )
TAKE( flagE , roomE )
TAKE( flagF , roomE )
MOVE( roomE , roomC )
MOVE( roomC , hallB )
MOVE( hallB , roomB )
TAKE( flagB , roomB )
MOVE( roomB , hallB )
MOVE( hallB , hallA )
MOVE( hallA , roomA )
TAKE( flagA , roomA )
MOVE( roomA , hallA )
MOVE( hallA , roomD )
TAKE( flagD , roomD )
```

Listing 4: Individual Plan for Agent 2 Starting in RoomE

```
TAKE( flagF , roomE )
TAKE( flagE , roomE )
MOVE( roomE , roomC )
TAKE( flagC , roomC )
MOVE( roomC , hallB )
MOVE( hallB , roomB )
TAKE( flagB , roomB )
MOVE( roomB , hallB )
MOVE( hallB , hallA )
MOVE( hallA , roomA )
TAKE( flagA , roomA )
MOVE( roomA , hallA )
MOVE( hallA , roomD )
TAKE( flagD , roomD )
```

Figure 3: Individual plan.

Listing 1: Joint-Plan for Agent 1 Starting in HallA

```
MOVE( hallA , roomA )
TAKE( flagA , roomA )
MOVE( roomA , hallA )
MOVE( hallA , hallB )
MOVE( hallB , roomB )
TAKE( flagB , roomB )
MOVE( roomB , hallB )
MOVE( hallB , hallA )
MOVE( hallA , roomD )
```

Listing 2: Joint-Plan for Agent 2 Starting in RoomE

```
TAKE( flagF , roomE )
TAKE( flagE , roomE )
MOVE( roomE , roomC )
TAKE( flagC , roomC )
MOVE( roomC , hallB )
MOVE( hallB , hallA )
MOVE( hallA , roomD )
TAKE( flagD , roomD )
```

Figure 4: Joint plan.

Each of these plans are translated into a sequence of states that the agent should follow (Figure 5).

```
0  robot-in_hallA
1  robot-in_roomA
2  robot-in_roomA  taken_flagA
3  robot-in_hallA  taken_flagA
4  robot-in_hallB  taken_flagA
5  robot-in_roomB  taken_flagA
6  robot-in_roomB  taken_flagA  taken_flagB
7  robot-in_hallB  taken_flagA  taken_flagB
8  robot-in_hallA  taken_flagA  taken_flagB
9  robot-in_roomD  taken_flagA  taken_flagB
```

Figure 5: State-Based Joint-Plan.

The agent's potential $\phi$ can be chosen as :

$$\phi(s) = \omega * CurrentStepInPlan$$
$$\omega = MaxReward/NumStepsInPlan$$

where $\omega$ is a scaling factor and $CurrentStepInPlan$ is the corresponding plan's step of the agent's state. We will compare the differences between individual and joint plan based reward shaping and we will see the benefits and constraints of each.

## Results and Discussion

### Initial results

After running thirty times all experiments, the mean discounted reward per episode were plotted on the following graphs. The discounted reward is the original reward multiplied by the discounted factor exponent the step number needed to reach the goal (Efthymiadis and Kudenko, 2013). Our initial case results are shown on Figure 6.

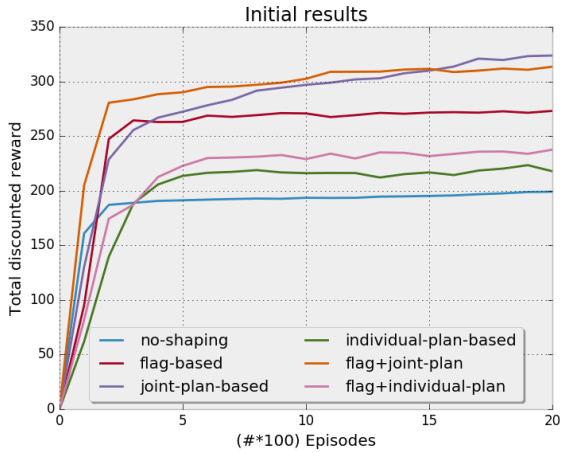


Figure 6: Initial results.

### Improved knowledge

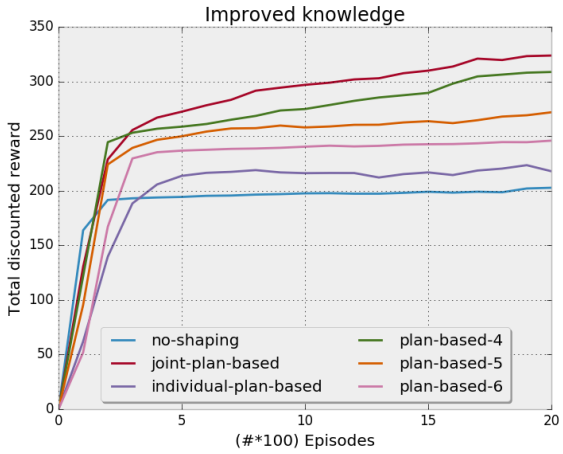


Figure 7: Improved knowledge.

### Improved cooperation

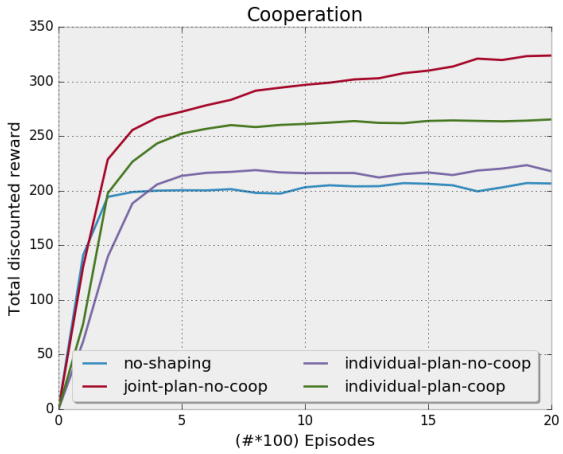


Figure 8: Improved cooperation.

## Conclusion

## References


Andrew Y. Ng, D. H. and Russell, S. J. (1999). Policy invariance under reward transformations : Theory and application to reward shaping. *ICML '99 Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287.

Devlin, S. and Kudenko, D. (2004). Plan-based reward shaping for multi-agent reinforcement learning. *The Knowledge Engineering Review*, 00:1–24.

Efthymiadis, K. and Kudenko, D. (2013). Using plan-based reward shaping to learn strategies in starcraft: Broodwar. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE.

Grzes, M. and Kudenko, D. (2011). *Reward Shaping and Mixed Resolution Function Approximation*. IGI Global.

Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158.