# Knowledge Revision for Reinforcement Learning with Abstract MDPs

Kyriakos Efthymiadis
Department of Computer Science
University of York, UK
ke517@york.ac.uk

Daniel Kudenko
Department of Computer Science
University of York, UK
daniel.kudenko@york.ac.uk

## ABSTRACT

Reward shaping is a method often used in RL so as to provide domain knowledge to agents and thus improve learning. An unrealistic assumption however is that the provided knowledge is always correct. This assumption can lead to poor performance in terms of total reward and convergence speed in case it is not met. Previous research demonstrated the use of plan-based reward shaping with knowledge revision in a single agent scenario where agents showed that they can quickly identify and revise erroneous knowledge and thus benefit from more accurate plans. This method however has no mechanism to deal with non-deterministic scenarios and is thus limited to deterministic domains. In this paper we present a method to provide heuristic knowledge via abstract MDPs, coupled with a revision algorithm to manage the cases where the provided domain knowledge is wrong. We show empirically that our method can efficiently revise erroneous knowledge even in the cases where the environment is non-deterministic and also removes the need for some of the assumptions present in plan-based reward shaping with knowledge revision.

## Categories and Subject Descriptors

Computing methodologies [**Artificial Intelligence**]: Learning

## General Terms

Experimentation

## Keywords

reinforcement learning; reward shaping; knowledge revision

## 1. INTRODUCTION

Reinforcement learning has been 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 process.

In earlier work on *knowledge-based reinforcement learning* [3, 7, 8] 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, expert knowledge is often of a heuristic nature and therefore may contain inaccuracies. Previous research shows that when an agent is provided with incorrect knowledge, there is a direct impact in performance which points to a clear need for knowledge revision methods [6].

Previously, basic belief revision operations were used in order to revise wrong knowledge when an agent was shaped using plan-based reward shaping [6]. The results were very promising with the agent quickly revising the wrong parts of the knowledge base and thus benefiting from more accurate shaping. However, that method included exhaustively searching in the environment to verify the parts of the knowledge base that were deemed erroneous which can impact performance in larger domains.

Moreover, in order for the agent to be able to search the environment it was allowed to 'teleport' between states. This required the agent to run only in simulation as backtracking or jumping among states cannot take place in a real world problem. In addition it is not possible for the plan-based revision method to handle a stochastic domain.

Marthi [10] proposed the use of abstract MDPs as a source of reward shaping in which an abstract high-level MDP of the environment is defined and solved using dynamic programming, e.g. value iteration. The resulting value function can then be used in order to shape the agent. Since this method relies on abstract MDPs to provide additional rewards and the environment probabilities are a vital part, it can be used to tackle stochastic environments.

In this paper we propose a method to revise knowledge in abstract MDPs which removes the need for some of the assumptions and problems presented in the revision method for plan-based reward shaping [6].

We compare our revision method to that developed for agents using plan-based reward shaping [6] and we demonstrate empirically that our agent can achieve similar perfor-

**Appears in:** *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*
Copyright © 2015, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

mance when the agents are provided with wrong knowledge in deterministic domains.

In addition we demonstrate empirically that our agent can overcome the problems posed by erroneous knowledge in a stochastic environment through the use of knowledge revision.

# 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 [14]. 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 [12].

A 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 a 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 [2].

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 [14]. After each real transition, $(s, a) \rightarrow (s', r)$, in the environment, it updates state-action values by the formula:

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

where $\alpha$ is the rate of learning and $\gamma$ 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 $\epsilon$, by choosing a random action or exploits its current knowledge, with probability $1 - \epsilon$, by choosing the highest value action for the current state [14].

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 $\lambda$ (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 [14].

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 [11, 13]. This concept can be represented by the following formula for the SARSA algorithm:

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

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 [13]. To deal with such problems, potential-based reward shaping was proposed [11] as the difference of some potential function $\Phi$ defined over a source $s$ and a destination state $s'$:

$$F(s, s') = \gamma\Phi(s') - \Phi(s) \quad (3)$$

where $\gamma$ 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 [11].

Wiewiora [16] 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 [4].

With a dynamic potential function [5], 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 [3, 13] but some attempts have been made to automate [9, 10] and semi-automate [8] 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 [7, 8], generates a potential function from prior knowledge provided by a domain expert represented as a high-level STRIPS plan.

The STRIPS plan is translated[1] into a state-based representation so that, whilst acting, an agent's current state can be mapped to a step in the plan as shown in Figure 1.

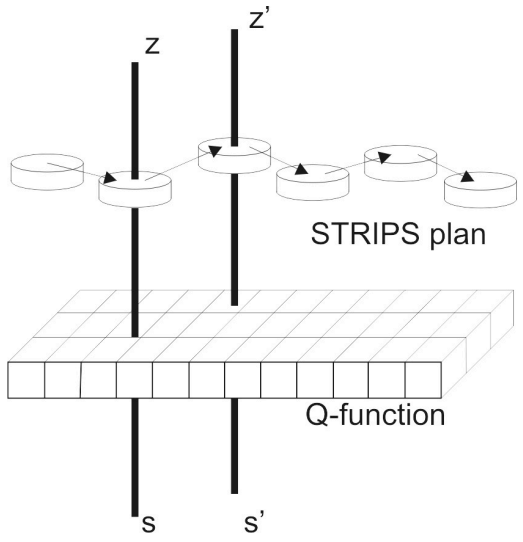



**Figure 1: Plan-Based Reward Shaping.**

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

$$\Phi(s) = CurrentStepInPlan * \omega \quad (4)$$

where $CurrentStepInPlan$ is the step number of the STRIPS plan that corresponds to state $s$ and $\omega$ is a scaling factor.

To not discourage exploration off the plan, if the current state is not in the state-based representation of the agent's plan then the potential used is that of the last state experienced that was in the plan. This feature of the potential function makes plan-based reward shaping an instance of dynamic potential-based reward shaping [5].

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.

## 2.4 Abstract MDP Reward Shaping

Plan-based reward shaping with knowledge revision has been shown to perform well in a deterministic environment [6]. However, while the results are promising this method is not able to handle stochastic environments since the domain dynamics cannot be captured by a STRIPS plan.

Marthi [10] proposed a general automatic framework to learn the potential function by solving an abstract MDP. The shaping algorithm obtains the potential function by solving a MDP which is based in an abstraction of the low level environment.

[1]This translation is automated by propagating and extracting the pre- and post- conditions of the high level actions through the plan.

Although the dynamics of the environment are often not known beforehand, the a MDP representing general abstract (and potentially heuristic) knowledge about the RL domain can be provided by domain experts before the learning process begins, similar to plan-based reward shaping.

In addition, by providing a mapping of low-level states of the environment to high-level abstract states[2], the abstract MDP can be solved using any off-the-shelf dynamic programming algorithm e.g. value iteration, before the main learning process begins, and the obtained value function can be used directly as a potential function of the low level environment:

$$\Phi(s) = V^*(z) * \omega, \quad (5)$$

where $V^*(z)$ is the optimal value function over the abstract state space $Z$ i.e. the space that contains all low level states $s$ which map to a high level state $z$ as shown in Figure 1, and $\omega$ is an optional scaling factor.

# 3. THE REVISION PROCESS

Providing expert domain knowledge to an agent can improve its learning performance. However, as mentioned previously, expert knowledge is often of a heuristic nature and therefore may contain inaccuracies. It has been shown that when an agent is guided by wrong knowledge it has a direct impact on its performance and can inhibit learning [6, 8].

A first successful attempt to tackle that problem was developed for agents using plan-based reward shaping in which an agent could revise potential errors in its knowledge base and thus benefit from more accurate shaping [6]. However that method required the agent to exhaustively search and verify wrong knowledge before deciding to revise. This can prove to be a daunting task especially in large domains.

Moreover, the agent needed to 'teleport' to any state while verifying wrong knowledge which is impractical or even infeasible in most real world scenarios and as mentioned previously, one of the assumptions in plan-based revision is the agent to run in simulation so as to make the 'teleportation' possible. Instead, our method does not require the agent to verify knowledge as the revision takes place alongside the agent's exploration of the low level states whilst learning.

Another issue was that there was no mechanism in place to tackle non-deterministic environments. The use of STRIPS does not allow the dynamics of the domain to be captured i.e. actions that can fail, probabilistic transitions etc. As a result, the agent would constantly revise parts of its knowledge back and forth due to the stochastic nature of the domain.

We propose using abstract MDPs with a revision algorithm which eliminates the above assumptions. The agent no longer needs to "teleport" since there is no need for verification. In addition, since this method is based on abstract MDPs, the stochastic domain dynamics can now be specified and thus the agent can tackle the problems of a non-deterministic environment.

In order to enable knowledge revision we allow the abstract MDP to constantly update its probabilities according to the agent's experiences in the low level environment using

[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.

**Algorithm 1** Revision Algorithm.

```
Solve abstract MDP
for episode = 0 to max_number_of_episodes do
  initialise transitions table T
  for step = 0 to max_number_of_steps do
    main learning process
    add transition to T
    if goal position then
      end episode
  /* add new transitions */
  for all transition in T do
    if transition not in abstract MDP then
      add transition to abstract MDP
      Pr(transition) = 1
  /* update the probabilities */
  for all transition in abstract MDP do
    if transition in T then
      Pr(transition)+ = α[1 − Pr(transition)]
    else
      Pr(transition)+ = α[0 − Pr(transition)]
  Solve abstract MDP
  /* continue to next episode */
```

the following formula (which is based on the computations of transition probabilities [15]):

$$Pr_{zz'} = \begin{cases} Pr_{zz'} + \alpha(1 - Pr_{zz'}) & \text{if } z, z' \text{ experienced} \\ Pr_{zz'} + \alpha(0 - Pr_{zz'}) & \text{otherwise} \end{cases} \quad (6)$$

In addition if the agent experiences a state transition which is not present in the abstract MDP, it is added with $Pr_{zz'} = 1$. The MDP is solved at the end of each episode and the new value function is used for shaping.

Equation 6 implies that if a transition from abstract state $z$ to $z'$ was performed during an episode then the probability of this transition is increased. If a certain transition from $z$ to $z'$ was not performed during an episode then the probability of this transition is lowered. This results in states which are not experienced, either because of wrong domain knowledge or because of the environment dynamics, to assume a lower probability based on Equation 6 which in turn results in a lower value $V(z)$.

Note that since the potential function changes whilst the agent is learning, this is an instance of dynamic potential-based reward shaping [5] for which the theoretical guarantees of policy invariance still hold.

# 4. EXPERIMENTAL DESIGN

We evaluate the revision algorithms using two domains. We use the flag collection domain from the work on plan-based reward shaping with knowledge revision to show that our method can reach a similar performance. In addition, we evaluate our algorithm in a non-deterministic Micro UAV problem, and demonstrate that our method can deal with both the stochastic nature of the domain, as well as erroneous knowledge provided to the agent. Note that this is a real-world domain provided by our industrial collaborators and shows the wider applicability of our method.

## *Flag Collection Domain*

The flag collection domain is an extended version of the navigation problem. The agent needs to navigate through the maze, collect flags and drop them off at a designated location.

At each time step it can decide to move north, east, south or west and will deterministically complete its move provided it does not collide with a wall which results in the agent staying at the same position. Regardless of the collected flags the episode ends once the agent reaches the goal position and the resulting reward is 100 times the number of collected flags. At each point the agent's state representation includes the position it is in the grid i.e. $x$ and $y$ coordinates and the flags it has picked up.

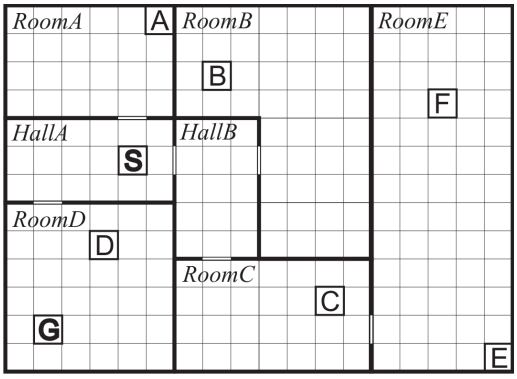



**Figure 2: Flag Collection Domain.**

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 agent to find where the flags are located.

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`, `S` is the starting position of the agent and `G` is the goal position.

Given this domain, the state abstraction includes the rooms and halls that the agent can navigate to i.e. `hallA, hallB, roomA, roomB` and so on as well as the location of flags e.g. `flagA_in_roomA`.

The action abstraction are the moves the agent can perform according to the layout e.g. `roomA_to_hallA`. The agent can also perform the action of picking a flag e.g. `taken_flagA`. By generating all the possible states and their respective probabilities an abstract MDP of the flag collection domain can be solved to be used as guidance by the agent.

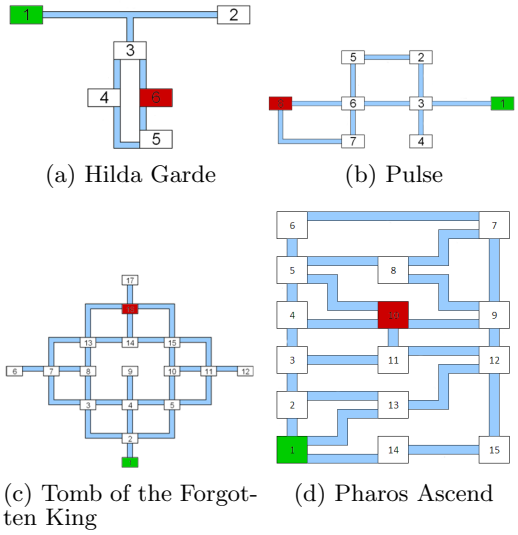
A partial example of the value function to be used for shaping is given in Listing 1 with $V(z)$ used in Equation 5 shown in the right hand column.

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

**Listing 1: Example Partial Value Function**

## *Micro UAV Problem*

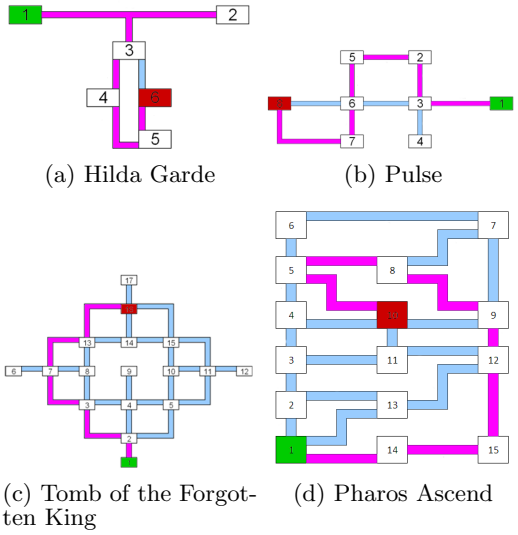
The Micro UAV problem is one where a micro UAV is intended to go through a building and locate a villain. This is realised as a grid world where the agent is provided with a high level map of the building it needs to search which serves as the domain knowledge that the agent is provided



(a) Hilda Garde (b) Pulse

(c) Tomb of the Forgotten King (d) Pharos Ascend

**Figure 3: Micro UAV Problem.**



(a) Hilda Garde (b) Pulse

(c) Tomb of the Forgotten King (d) Pharos Ascend

**Figure 4: Micro UAV Problem. Safe routes.**

with. The state representation for the RL agent is its position within the grid i.e. the $x$ and $y$ coordinates.

The building is comprised of rooms and corridors each associated with a probability that the agent gets caught by the enemy. All rooms are a $20x20$ grid and all corridors have a width of 5 squares. The domain knowledge that is provided is similar to that in the flag collection domain with rooms and halls but this time the knowledge also includes the probabilities that the agent might get caught at certain parts in the maze.

Specifically, the state abstraction for the abstract MDP includes rooms `1-17` and the probability of the agent being detected in those areas, halls `A-V` along with the probability of detecting the agent and the location of the villain within the building.

A partial abstract value function for this domain is shown in Listing 2 where the low values of certain states correspond to areas of high detection in the building and should be avoided.

```
robot_in(hallA) 30
robot_in(room1) 35
robot_in(hallC) 39
robot_in(room3) 10
robot_in(hallE) 8
robot_in(room5) 40
robot_in(hallB) 34
robot_in(hallD) 4
```

**Listing 2: Partial Abstract Value Function in the Micro UAV Domain**

Within the low level grid the agent can choose to move north, east, south or west to one of its neighbouring squares. If the agent is caught while searching the building then the episode ends and the agent receives a reward of $-50$. If the agent manages to successfully locate the villain i.e. enter the square the villain is located, it is given a reward of 1000 and the episode is reset.

It is worth noting that a safe path to the villain always exists i.e. a path where there is a low probability of the agent being detected as shown in Figure 4. The rooms that are not part of this safe route have a $0.4 - 0.6$ probability of the agent being detected while the corridors a $0.7 - 0.9$ probability.

Figure 3 shows the map configuration used for evaluating this method in the Micro UAV problem. The room the agent is located at the start of an episode is shown in green and the villain's position within the building is shown in red.

## 4.1 Results

### *Flag Collection Domain*

Our agent is compared against an agent with knowledge revision using plan-based reward shaping. The comparison is based on the performance in terms of discounted goal reward.

All agents implemented SARSA with $\epsilon-$greedy action selection and eligibility traces[3] [14]. 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 30 iterations each lasting $50,000$ episodes.

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. The graphs also include error bars showing the standard error of the mean. We compare the agents across three classes of erroneous knowledge:

1) Incorrect knowledge where the agents are provided with knowledge which contains extra information which is not present in the environment. This means that the knowledge contains those facts that are correct in the environment as well as more irrelevant information. In this case the knowledge contains 2 to 6 extra flags which are not present in the

[3]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 [1]. Furthermore, it has been shown to work before without eligibility traces [3, 11].

simulation but also includes those that are in the correct position[4].

2) Incomplete knowledge where the agents are provided with knowledge which is missing important goals. In this case 1 to 3 flags are missing from the provided knowledge and thus the shaping does not provide an incentive to collect them.

3) Combination of incorrect and incomplete knowledge where the provided knowledge not only contains incorrect and incomplete knowledge in terms of flags as described previously, but also wrong connections between rooms and misplaced flags. Since the erroneous parts of the knowledge are generated randomly in each experiment there can be cases where the knowledge does not contain any correct facts i.e. all the transitions are either missing or do not have the correct probabilities. We present these results only for the abstract MDP shaping with revision as the plan-based method cannot handle these types or wrong knowledge efficiently.

The results are shown in Figures 5, 6 and 7 presenting the average performance over 30 runs, each run involving different instances of erroneous knowledge chosen at random as described previously.

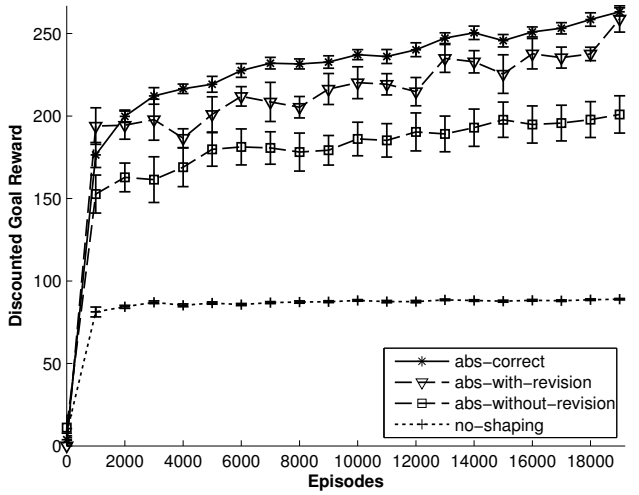



**Figure 5: Abstract MDP agent with combination of erroneous knowledge.**

The biggest performance improvement of our revision algorithm is shown in the combination of incorrect and incomplete knowledge shown in Figure 5. The agent there has to deal with potentially, entirely wrong knowledge in terms of transitions between rooms, missing flags, extra flags and misplaced flags as mentioned earlier. However the results show that by updating the probabilities of the states in the abstract MDP and by adding new transitions as it encounters them, it manages to identify the wrong parts in the knowledge which are revised out, while including the correct states and transitions to its knowledge. This results in the agent achieving a similar performance to the agent receiving correct knowledge.

In the incomplete knowledge case, the agent can encounter states which are not in its provided knowledge. Our agent

[4]More experiments have been conducted with varying errors in the knowledge base and the results show the same trend as the results presented here.

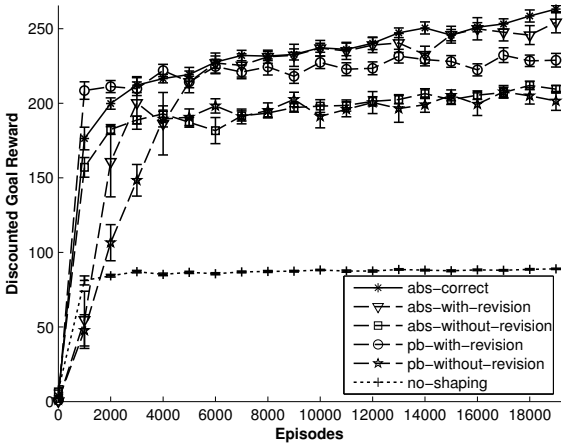



**Figure 6: Incomplete knowledge comparison.**

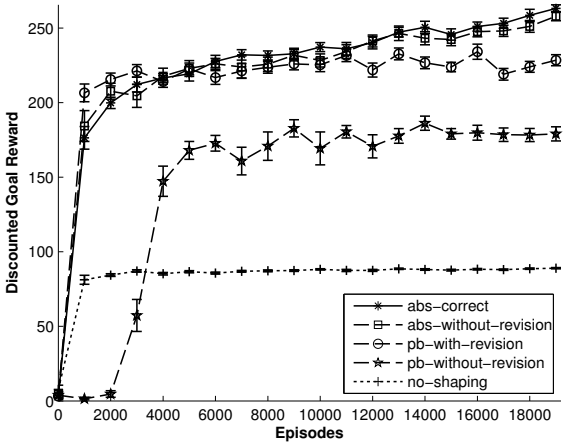



**Figure 7: Incorrect knowledge comparison.**

manages to quickly identify the parts which are missing from the provided MDP. By adding these new transitions it encounters in the low-level to the abstract MDP, it manages to solve a more accurate MDP and thus benefit from better shaping.

The results are interesting when it comes to the "incorrect" case of wrong knowledge in abstract MDPs. While the plan-based agent has an impact on behaviour and a need for revision is apparent, this is not the case for the abstract MDP agent. It appears that the agent's performance is not impacted at all[5].

Taking a closer look at how the abstract MDP provides extra rewards, it becomes clear why this happens. Since a value function is used as a reward shaping source, every state the agent finds itself in will have a potential that will lead to the goal. These multiple paths to the goal mean that the agent will never be left without guidance. Paths which are not encountered by the agent because they do not exist do not feature at all when receiving rewards. Therefore there is

[5]Due to this behaviour, we have not included the abstract MDP agent with knowledge revision to Figure 7 since it achieves a similar performance to the agent without revision and the agent with correct knowledge.

no need to revise incorrect knowledge when using abstract MDP shaping. However, it is this specific reward shaping method that can handle incorrect knowledge and as shown in Figures 6 and 5, a need for revision is still apparent when the agent is given incomplete knowledge, or a combination of incorrect and incomplete knowledge.

In contrast, the plan-based agent receives only a single path to the goal. Therefore if the agent cannot achieve a step in the provided plan because the path does not exist, it does not receive any further guidance after that point and is effectively left to act without reward shaping.

### *Micro UAV Problem*

While our abstract MDP method works perfectly well in a deterministic domain, in order for it to be widely applicable it needs to be able to tackle the complexity of non-determinism. Therefore, we evaluate our method in a non-deterministic domain, the Micro UAV problem which was presented previously. Plan-based reward shaping with knowledge revision cannot tackle such a domain as there is no mechanism in place to account for non-deterministic environments.

We have used the same parameters as those used in the flag collection domain to conduct these experiments. For clarity we only show up to 3000 episodes since there was no change in performance after that point. The graphs also include error bars showing the standard error from the mean.

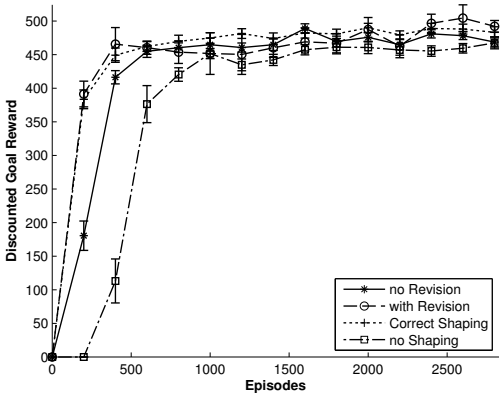



**Figure 8: Hilda Garde.**

The erroneous knowledge the agents are given is both incorrect and incomplete with varying degrees of "wrongness" per experiment i.e. the agent receives a combination of erroneous knowledge which can vary for each experiment in the number of incorrect or incomplete parts regarding rooms, corridors and their connections, villain location and transition probabilities. The number of incorrect and incomplete parts can vary from 2 up to 5 and are generated randomly for each experiment.

For example, in the Hilda Garde building shown in Figure 3 the agent can receive domain knowledge which can be missing `room3`, have additional information of a corridor, `hallT`, connecting `room5` to `room6` and also have the probability that the agent gets detected in the corridors set to 0 for all corridors.

A sample value function for this example is shown in Listing 3. The agent will need to utilise its knowledge revision

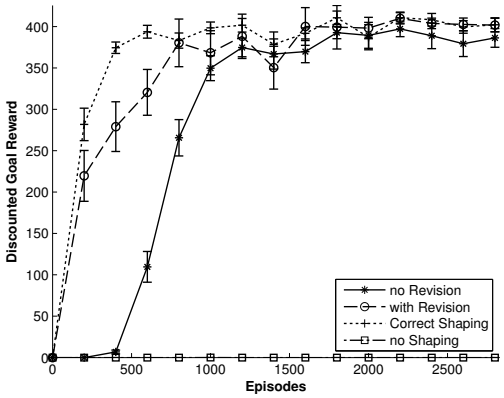



**Figure 9: Pulse.**

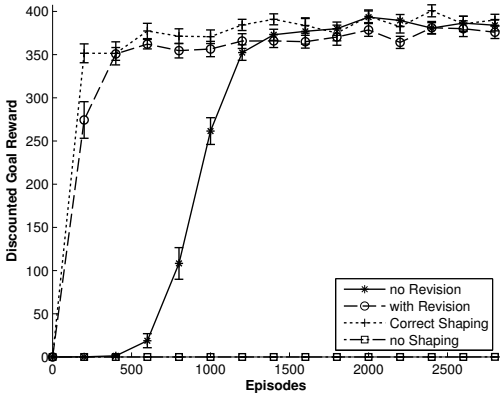



**Figure 10: Tomb of the Forgotten King.**

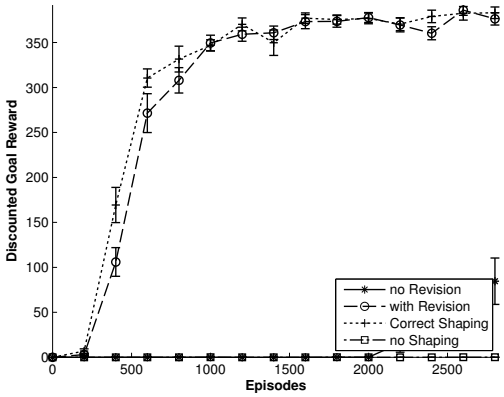



**Figure 11: Pharos Ascend.**

capabilities and overcome the problems posed by this erroneous knowledge. Figures 8, 9, 10 and 11 show the agents' performance.

Similar to the flag collection domain the abstract MDP agent can efficiently revise erroneous knowledge even when dealing with a non-deterministic environment. On all configurations the agent manages to quickly overcome the problems posed by erroneous knowledge and reach a performance similar to the agent receiving correct knowledge.

```
robot_in(room1) 0
robot_in(hallA) 0
robot_in(room2) 0
robot_in(hallB) 46
robot_in(room4) 55
robot_in(hallC) 68
robot_in(room5) 75
robot_in(hallD) 85
robot_in(hallT) 85
robot_in(room6) 100
robot_in(hallE) 81
```

**Listing 3: Sample Erroneous Value Function in the Hilda Garde Domain**

The agent without knowledge revision takes longer to find the optimal policy, around 1000 episodes on most cases apart from Pharos Ascend where it takes more than 4000 episodes to start improving, while the agent receiving no shaping is not able to overcome the problem of non-determinism and cannot reach a good enough policy within the time frame shown in the graphs.

As the theory suggests all agents are able to reach the same performance after episode 15000 but the benefits of using reward shaping and knowledge revision are visible especially at the start of the experiments.

# 5. CONCLUSION

Imparting domain knowledge through reward shaping can significantly increase the learning performance of a RL agent. However, the provided knowledge may be erroneous which results in a need for revision. Previous research showed promising results in deterministic domains but stochastic environments could not be handled properly [6].

In this paper, we propose a revision algorithm for agents using abstract MDP reward shaping. We demonstrate empirically that our approach can achieve a similar behaviour to the revision method using plan-based reward shaping in deterministic environments, while removing the need for certain assumptions which make the plan-based method impossible to use in non-deterministic environments.

More importantly, we apply our method to a non-deterministic environment and empirically demonstrate that our agent manages to efficiently revise erroneous knowledge in order to reach better policies, quickly reaching performance of an agent that receives the correct knowledge.

# 6. REFERENCES


[1] J. Asmuth, M. Littman, and R. Zinkov. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 604–609, 2008.

[2] D. P. Bertsekas. *Dynamic Programming and Optimal Control (2 Vol Set)*. Athena Scientific, 3rd edition, 2007.

[3] S. Devlin, M. Grześ, and D. Kudenko. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 2011.

[4] S. Devlin and D. Kudenko. 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.

[5] S. Devlin and D. Kudenko. Dynamic potential-based reward shaping. In *Proceedings of The Eleventh Annual International Conference on Autonomous Agents and Multiagent Systems*, 2012.

[6] K. Efthymiadis, S. Devlin, and D. Kudenko. Overcoming erroneous domain knowledge in plan-based reward shaping. In *Autonomous agents and multi-agent systems*, pages 1245–1246. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[7] K. Efthymiadis and D. Kudenko. Using plan-based reward shaping to learn strategies in starcraft: Broodwar. In *Computational Intelligence and Games (CIG)*. IEEE, 2013.

[8] M. Grześ and D. Kudenko. Plan-based reward shaping for reinforcement learning.

[9] M. Grześ and D. Kudenko. Multigrid Reinforcement Learning with Reward Shaping. *Artificial Neural Networks-ICANN 2008*, pages 357–366, 2008.

[10] B. Marthi. Automatic shaping and decomposition of reward functions. In *International Conference on Machine learning*, page 608. ACM, 2007.

[11] A. Y. Ng, D. Harada, and S. J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pages 278–287, 1999.

[12] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., New York, NY, USA, 1994.

[13] J. Randløv and P. Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the 15th International Conference on Machine Learning*, pages 463–471, 1998.

[14] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[15] C. Szepesvári and M. L. Littman. Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms. *Proceedings of International Conference of Machine Learning*, 96, 1996.

[16] E. Wiewiora. Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19(1):205–208, 2003.