What If the World Were Different? Gradient-based Exploration for New Optimal Policies

Rui Silva[1], Francisco S. Melo[2], and Manuela Veloso[3]

[1] Computer Science Department, Carnegie Mellon University, Pittsburgh PA, USA
INESC-ID / Instituto Superior Técnico, Universidade de Lisboa, Portugal
ruisilva@cmu.edu
[2] INESC-ID / Instituto Superior Técnico, Universidade de Lisboa, Portugal
fmelo@inesc-id.pt
[3] Machine Learning Department, Carnegie Mellon University, Pittsburgh PA, USA
mmv@cs.cmu.edu

Abstract

Planning under uncertainty assumes a model of the world that specifies the probabilistic effects of the actions of an agent in terms of changes of the state. Given such model, planning proceeds to determine a policy that defines for each state the choice of action that the agent should follow in order to maximize a reward function. In this work, we realize that the world can be changed in more ways than those possible by the execution of the agent's repertoire of actions. These additional configurations of the world may allow new policies that let the agent accumulate even more reward than that possible by following the optimal policy of the original world. We introduce and formalize the problem of planning while considering these additional possible worlds. We then present an approach that models feasible changes to the world as modifications to the probability transition function, and show that the problem of computing the configuration of the world that allows the most rewarding optimal policy can be formulated as a constrained optimization problem. Finally, we contribute a gradient-based algorithm for solving this optimization problem. Experimental evaluation shows the effectiveness of our approach in multiple problems of practical interest.

1 Introduction

Consider the following situation. A mobile assistant robot docked at its charging station (located in A) receives a request from a user to pick up a package from the user's office (located in B). For illustrative purposes, let us consider the environment layout depicted in Fig. 1a. In order to complete this task, the robot must plan the necessary course of action that leads from A to B, preferably in the most efficient way. Given the environment layout in Fig. 1a, the most efficient plan is to travel from A all the way to the end of the corridor, move down, and then travel left, all the way until point B, as depicted in Fig. 1b.

Suppose, now, that there is actually a door between locations A and B in the environment, a door that the robot is unable to open and which is usually to remain shut. If that door were

open, the robot could quickly reach location B and attend to the user's request. The user may even be willing to open the door for the robot to go through, if that resulted in an advantage to the user—the user's request is attended to sooner. However, since the user in general may not know where the robot is and the door is expected to remain shut, the user will not open the door for the robot of its own initiative.

The example above illustrates the key problem addressed in this paper. Planning approaches usually assume a model of the interaction between the agent and the world that describes how the actions of the agent impact the state of the world. However, recent work—particularly in the robotics community—has contemplated the situation in which the agent/robot reaches out to human users, enlisting their assistance in order to render the robot's task easier to complete [18, 11, 17, 14] or even to recover from failures [12]. Such assistance may consist in modifying the state of the world [18, 19, 12, 11], to perform actions that the robot cannot [17, 14] or effectively modifying the layout of the environment [8, 7]. Other works have looked into the problem of explaining why a solution to a planning problem could not be found, proposing different world configurations in which a solution does exist [9, 6].

In this paper, we propose a general model in which the agent/robot is allowed to explicitly reason, at a "meta level", about possible configurations of the environment. In particular, we propose that such reach-out actions—in which the agent enlists the assistance of a human user—translate into changes to the dynamic model of the world. By reasoning about the impact of these changes in the agent's behavior towards some target task, the agent can not only enlist the user's assistance, but also guide the user's actions towards the world configuration that is most convenient, in light of the task.

We adopt a decision-theoretic framework, in which the task the agent must complete is described by a Markov decision process, and the different world configurations translate in different transition probabilities. In this setting, a "better" world configuration corresponds to a transition probability matrix for which the corresponding optimal policy yields a larger total discounted reward. We formulate the problem of reasoning over "good" world configurations as a non-convex constrained optimization problem, in which the agent explicitly balances the benefits arising from changing the world with the potential costs incurred in such change. We also propose a novel gradient-based approach to solve such optimization problem.

Summarizing, our contributions in this paper are two-fold.

- We contribute a novel formalization of the problem of reasoning over world configurations as a constrained optimization problem that balances the benefits associated with different world configurations with the costs associated with such configurations.
- We propose a gradient-based algorithm that effectively addresses the associated optimization problem.

We illustrate the applicability of our approach on several scenarios of practical interest, including the taxi domain [4].

(a) (b)

Figure 1: Illustration of the corridor scenario, where a mobile robot departs from its docking station at A to attend a user request at B. 1a shows the world configuration. 1b depicts the optimal policy, which requires the robot to travel all the way to the end of the upmost corridor, before moving down and traveling all the way back to B

2 Background

A Markov decision process (MDP) is a tuple $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$, describing a sequential decision problem under uncertainty. $\mathcal{X}$ is the set of possible states of the world and $\mathcal{A}$ the set of actions available to the agent. When the agent takes an action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$, the world transitions to state $y \in \mathcal{X}$ with probability $P(y \mid x, a)$ and the agent receives a reward $r(x, a)$. The discount factor $\gamma \in [0, 1)$ determines the relative importance of present and future rewards. The goal of the agent is to identify a *policy*—i.e., a mapping π such that $\pi(a \mid x)$ is the probability of selecting action $a \in \mathcal{A}$ when in state $x \in \mathcal{X}$—such that the value

$$v^\pi(x) \triangleq \mathbb{E}_{a_t \sim \pi(x_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \mid x_0 = x \right]$$

is maximized for all states $x \in \mathcal{X}$. The vector $\boldsymbol{v}^\pi$ is called the *value function* associated with policy π and can be computed as the solution to the linear system

$$\boldsymbol{v}^\pi = \boldsymbol{r}^\pi + \gamma \mathbf{P}^\pi \boldsymbol{v}^\pi, \tag{1}$$

where $\boldsymbol{r}^\pi$ and $\mathbf{P}^\pi$ are such that

$$[\boldsymbol{r}^\pi]_x = \sum_{a \in \mathcal{A}} \pi(a \mid x) r(x, a)$$
$$[\mathbf{P}^\pi]_{xy} = \sum_{a \in \mathcal{A}} \pi(a \mid x) P(y \mid x, a).$$

Specifically,

$$\boldsymbol{v}^\pi = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \boldsymbol{r}^\pi, \tag{2}$$

where $\mathbf{I}$ is the $|\mathcal{X}| \times |\mathcal{X}|$ identity matrix. It is a well-known fact that the matrix $(\mathbf{I} - \gamma \mathbf{P}^\pi)$ is non-singular [16] and, therefore, the solution to (1) is well-defined. Solving an MDP thus consists of computing a policy π^* such that, for all $x \in \mathcal{X}$ and all policies π,

$$v^{\pi^*}(x) \geq v^\pi(x),$$

which can be done using dynamic programming, linear programming, stochastic approximation, among other approaches.

3 Problem Formulation

Consider an agent facing a sequential decision problem described as an MDP $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$, where the initial state, x_0, follows some known distribution μ_0. In the MDP model, it is possible to associate the reward function r to the *goal* of the agent, and the transition probabilities P with the *dynamics* of the world. Following our discussion in Section 1, we are interested in investigating how different world configurations may impact the decision problem faced by the agent. In practice, different world configurations mostly impact the way the agent interacts with the world (i.e., the outcome of the agent's actions) which, in the MDP model, is described by the transition probabilities, P. As such, we describe different world configurations in terms of different transition probabilities. Formally, we write P_θ to denote the transition probabilities associated with the world configuration θ, and write Θ to denote the set of all possible configurations.

Note that this set need not be countable, as the set of possible configurations may be continuous. Note also that each configuration θ induces a different MDP $(\mathcal{X}, \mathcal{A}, P_\theta, r, \gamma)$, and we denote the corresponding optimal policy by π^*_θ.

To allow the agent to reason about the benefits of different world configurations and corresponding optimal policies, we let $J(\theta)$ denote the *value* associated with world configuration θ:

$$J(\theta) = \sum_{x \in \mathcal{X}} \mu_0(x) v^*_\theta(x), \tag{3}$$

where $v^*_\theta(x)$ denotes the value of policy π^*_θ in state $x \in \mathcal{X}$. Similarly, we let $C(\theta)$ denote the *cost* associated with the world configuration θ. Such cost translates the effort involved in shifting the world from the original configuration θ_0 to configuration θ. As discussed before, shifting the world may require reaching out to an external entity.

We can now define the problem of finding the "best" world configuration as the solution to the constrained optimization problem

$$\max_{\theta \in \Theta} F(\theta) = J(\theta) - C(\theta). \tag{4}$$

Several observations are in order. First of all, in spite of its apparent simplicity, (4) is a constrained non-convex optimization problem, since computing $\boldsymbol{v}^*_\theta$ involves solving the nonlinear recursion

$$v^*_\theta(x) = \max_{a \in \mathcal{A}} \left[r(x,a) + \gamma \sum_{y \in \mathcal{X}} P_\theta(y \mid x, a) v^*_\theta(y) \right]. \tag{5}$$

The dependence of the objective function F on θ is, therefore, non-trivial to express and optimize. We postpone to Section 4 a discussion on how the optimization problem in (4) can be efficiently tackled.

A second observation is that the formulation in (4) is rather general, as it does not rely on any particular parameterization of the transition probabilities. Such flexibility provides interesting bridges between the problem addressed here and other areas of research within the MDP literature—including cases in which the MDP model is not fully specified. We refer to Section 5.3 for a more detailed discussion of such connections.

We conclude this section by illustrating the use of our proposed framework in the simple corridor problem introduced in Section 1, highlighting several of its important features.

(a) 2×2 corridor scenario. (b) Objective function for $\gamma = 0.9$.

Figure 2: Layout and objective function for the corridor scenario from the example.

3.1 The corridor example

Consider once again the problem faced by a mobile robot that must navigate from location A to location B in the grid world depicted in Fig. 2a, where the obstacle between A and B actually corresponds to a door that is usually to remain shut. This problem can be described as an MDP $(\mathcal{X}, \mathcal{A}, P_0, r, \gamma)$, where $\mathcal{X} = \{A, X_1, B, X_2\}$ and $\mathcal{A} = \{u, d, l, r, s\}$, corresponding to the actions "Up", "Down", "Left", "Right" and "Stay", respectively. Each action moves the agent deterministically to the adjacent cell in the corresponding direction, except if an obstacle is found. The reward is -1 for all state-action pairs except (B, s), where it is 0. The initial distribution μ_0, in this case, is given by $\mu_0(x) = \mathbb{I}(x = A)$, where $\mathbb{I}(U)$ denotes the indicator for U, taking value 1 when U holds and 0 otherwise.

To apply our proposed formalization to the navigation scenario just described, we first identify the set of possible world configurations. In this case, for simplicity of analysis, we consider that $\Theta = [0, 1]$, where configuration θ corresponds to how much a door is opened ($\theta = 0$ corresponds to a fully closed door, while $\theta = 1$ corresponds to a fully open door). We consider that the probability of the robot successfully going through the door in either direction is proportional to θ.

Our configuration space Θ can be translated directly into a parameterization for the transition probabilities, where

$$P_\theta(B \mid A, d) = P_\theta(A \mid B, u) = \theta;$$
$$P_\theta(A \mid A, d) = P_\theta(B \mid B, u) = 1 - \theta.$$

Considering the simplest case where there is no cost for changing the world, *i.e.*, $C(\theta) \equiv 0$, the optimization problem in (4) can be simplified to

$$\max_{\theta \in [0,1]} J(\theta),$$

with

$$J(\theta) = v^*_\theta(A) = \max\left\{-1 - \gamma - \gamma^2, -\frac{1}{1 - \gamma(1 - \theta)}\right\}. \tag{6}$$

The objective function is plotted in Fig. 2b for $\gamma = 0.9$. As expected, the maximum value is obtained when $\theta = 1$, which leads to a policy value of -1 in contrast with the value of -2.7 obtained by the optimal policy in the original environment.

Another important observation is that, as expected from our previous discussion, the objective function is non-convex in θ, which renders the optimization problem in (4) hard to solve, in general.

4 Approach

As discussed in Section 3, the ability of the agent to reason about alternative world configurations towards better optimal policies for some target task was formalized as the constrained optimization problem in (4). Unfortunately, the corresponding objective function is, in general, non-convex and, therefore, potentially hard to solve exactly. Instead, we propose an iterative, gradient-based algorithm that we dub P-ITERATION.

Specifically, our approach departs from an arbitrary initial configuration $\theta^{(0)}$ and iteratively builds a sequence $\left\{\theta^{(k)}, k = 1, \ldots\right\}$ that will converge to a local maximum of $F(\theta) = J(\theta) - C(\theta)$ as

$$\theta^{(k+1)} = \text{Proj}_{\Theta}[\theta^{(k)} + \alpha \nabla_{\theta} F(\theta^{(k)})],$$

where Proj_{Θ} is a projection operator that ensures that the iterates of the algorithm remain within the set Θ of admissible configurations. However, in light of the non-trivial dependence of J on θ, the computation of the gradient $\nabla_{\theta} F(\theta)$ is also non-trivial. We now detail how such gradient can be computed.

4.1 Differentiation of $J(\theta)$

We start by observing that

$$\nabla_{\theta} F(\theta) = \nabla_{\theta} J(\theta) - \nabla_{\theta} C(\theta).$$

Although we do not specify how the cost function depends on θ (it is problem-specific), we will admit that such dependence can be formulated in such a way that the term $\nabla_{\theta} C(\theta)$ can easily be computed. As for the term $\nabla_{\theta} J(\theta)$, using vector notation, we get

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} [\boldsymbol{\mu}_0^{\top} \boldsymbol{v}_{\theta}^*],$$

where $\boldsymbol{\mu}_0$ is an $\mathcal{X}$-dimensional vector with xth component given by $\mu_0(x)$ and $^{\top}$ is the transpose operation. Since $\boldsymbol{\mu}_0$ does not depend on θ, we only have to compute $\nabla_{\theta} \boldsymbol{v}_{\theta}^*$. Unfortunately, as seen in Section 3, $\boldsymbol{v}_{\theta}^*$ is the solution of the fixed-point equation in (5), and the derivative cannot be directly computed.

Instead, we note that, for small changes in θ, the optimal policy remains unchanged, except eventually in a small subset of θ. [1] On the other hand, from (2),

$$\boldsymbol{v}_{\theta}^* = (\mathbf{I} - \gamma \mathbf{P}_{\theta}^{\pi_{\theta}^*})^{-1} \boldsymbol{r}^{\pi_{\theta}^*}.$$

Therefore, by fixing π_{θ}^*, we finally get

$$\nabla_{\theta} \boldsymbol{v}_{\theta}^* \approx \gamma (\mathbf{I} - \gamma \mathbf{P}_{\theta}^{\pi_{\theta}^*})^{-1} \nabla_{\theta} \mathbf{P}_{\theta}^{\pi_{\theta}^*} \boldsymbol{v}_{\theta}^*.$$

4.2 P-ITERATION

P-ITERATION is summarized in Algorithm 1. It includes an outer cycle that performs N random restarts, to avoid local maxima. Within such outer cycle, the algorithm merely performs standard projected gradient updates until a stopping condition is met, usually involving either the norm of the gradient or the distance between consecutive estimates.

[1] We do not establish such fact in this paper, but it can be established by replicating the proof of Proposition 4 in [13].

Algorithm 1 P-ITERATION algorithm

Require: MDP, $\mathcal{M} = (\mathcal{X}, \mathcal{A}, P_{\theta_0}, r, \gamma)$
Require: Initial state distribution, μ_0
Require: Configuration space, Θ
Require: Cost function, C
Require: Number of restarts, N
Require: Stopping condition, ϵ

$\theta^* = \theta_0$
$F^* = J(\theta^*) - C(\theta^*)$
for $n = 1$ **to** N **do**
  $k \leftarrow 0$
  Randomly select $\theta^{(0)}$
  repeat
    $\boldsymbol{v}^*_{\theta^{(k)}}$ using (5)
    $J(\theta^{(k)}) \leftarrow \boldsymbol{\mu}_0^\top \boldsymbol{v}^*_{\theta^{(k)}}$
    $\nabla_\theta F(\theta^{(k)}) = \nabla_\theta J(\theta^{(k)}) - \nabla_\theta C(\theta^{(k)})$
    Update

$$\theta^{(k+1)} \leftarrow \text{Proj}_\Theta[\theta^{(k)} + \alpha \nabla_\theta F(\theta^{(k)})]$$

    $k \leftarrow k + 1$
  until $\left\|\theta^{(k)} - \theta^{(k-1)}\right\| < \epsilon$
  $F^{(n)} = J(\theta^{(k)}) - C(\theta^{(k)})$
  if $F^{(n)} > F^*$ **then**
    $\theta^* = \theta^{(k)}$
    $F^* = F^{(n)}$
return θ^*

5 Results

We evaluate and analyze the performance of the P-ITERATION algorithm in a number of different scenarios. These scenarios were selected to enable an analysis of P-ITERATION from different perspectives. We start by considering a set of navigation scenarios that can be easily parameterized in terms of size. This facilitates an analysis of the scalability of our approach, both in terms of execution time and quality of solutions achieved. Additionally, these navigation scenarios are particularly well structured, allowing an intuitive interpretation of the resulting solutions. Besides the navigation scenarios, we also test our approach in random scenarios and a real-world robotic water pouring task. These tasks demonstrate the applicability of our method in scenarios where the world configurations lead to complex parameterizations of the transition probabilities.

5.1 Navigation scenarios

We start by analyzing the performance of our approach in more complex versions of the corridor example from Section 3. We consider three types of navigation scenarios:

- The corridor scenarios, consisting of $2 \times N$ grids in which the top and bottom rows are separated by an obstacle except in the Nth column (see Fig. 1 for $N = 4$);

- The maze scenarios, consisting of a $N \times N$ grid in which every pair of adjacent rows is separated by obstacles except in one of the ends (see Fig. 3c for $N = 4$).
- The taxi scenario is a benchmark problem from the reinforcement learning literature [4], and is depicted in Fig. 3d. In the taxi domain, an agent must navigate a 5×5 maze-like environment, picking up and dropping off a passenger from/to a set of pre-designated locations (marked as R, G, Y and B in the diagram).

All navigation scenarios are modeled as MDPs in which the state-space corresponds to the possible positions of the agent; in the taxi domain, the state space also includes the location of the passenger (in one of the marked sites or in the taxi) and its destination (one of the marked sites). The action space includes the four movement actions featured in the example of Section 3; the corridor and maze domains also include the action "stay", while the taxi domain additionally includes the actions "pick-up" and "drop". In the base environment configuration, all transitions are deterministic.

In all domains, the reward function penalizes the agent with -1 for every navigation action. Additionally, in the corridor and maze domains, the reward is 0 for executing action "stay" in the goal state (marked with B); in the taxi domain, the reward for a successful drop-off is 20, while an unsuccessful drop-off (in the wrong location) penalizes the agent with -10.

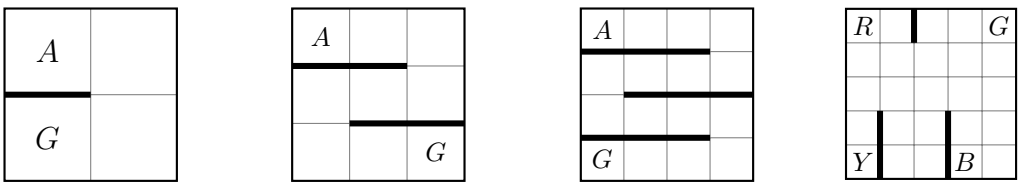
In the corridor scenarios, a configuration $\boldsymbol{\theta}$ is a vector in $\Theta = [0, 1]^K$, where we admit that K obstacles correspond to "doors", and θ_k indicates how much the kth door in the corridor is open. Similarly, in the maze scenarios, a configuration $\boldsymbol{\theta}$ is a vector in $\Theta = [0, 1]^{N-1}$, where we admit that all obstacles adjacent to an environment border correspond to "doors", and θ_k indicates how much the door in the kth row is open. Finally, in the taxi scenario, a configuration $\boldsymbol{\theta}$ is a vector in $\Theta = [0, 1]^K$, where we admit that K of the obstacles in the environment correspond to "doors", and θ_k indicates how much the kth door is open. The parameterization of the transition probabilities, in these scenarios, is similar to the one discussed in the example from Section 3.

Finally, we take μ_0 as the uniform distribution and

$$C(\theta) = \frac{1}{|\mathcal{X}|} \sum_{k=1}^{K} \left(\frac{2}{1 + e^{-\beta\theta_k}} - 1 \right), \tag{7}$$

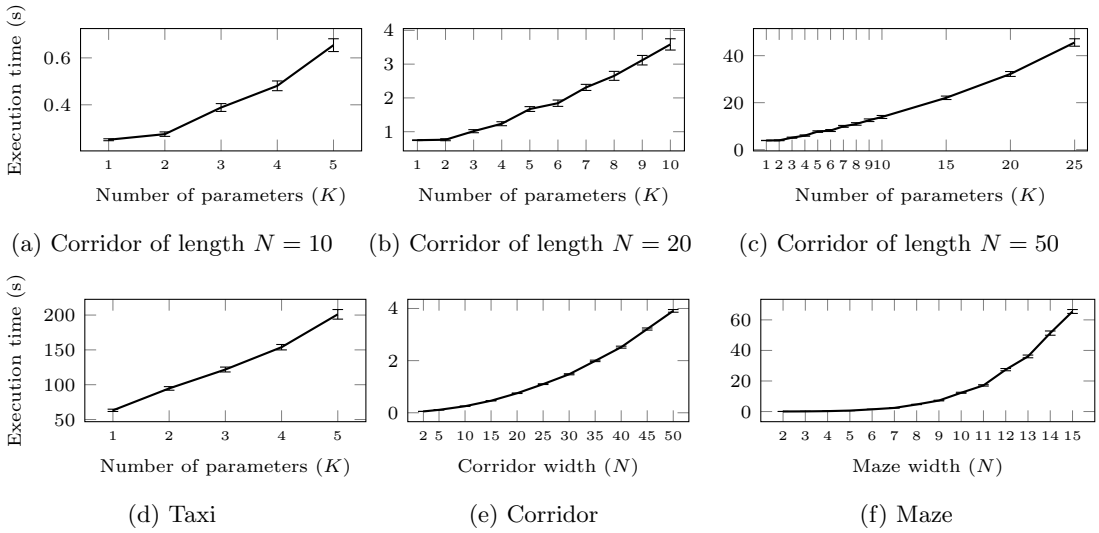
with $\beta = 100$. This cost function is a smooth approximation of a step-size function in each component of θ.

Figure 4 and Table 1 summarize the performance of P-ITERATION in the different navigation scenarios, for different values of K and problem sizes. The former depicts the performance in terms of execution time. The latter in terms of the quality of solutions achieved. We ran each scenario for 50 random restarts, and used the Adam gradient algorithm as described in [10]. All results are averaged over 30 randomly seeded runs.



(a) Maze scenario ($N = 2$) (b) Maze scenario ($N = 3$) (c) Maze scenario ($N = 4$) (d) Taxi scenario.

Figure 3: Example navigation scenarios.



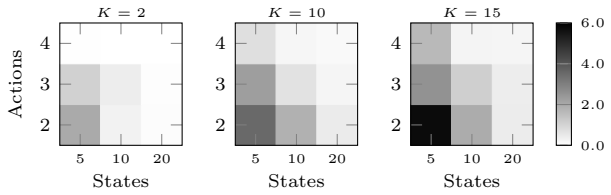
(a) Corridor of length $N = 10$ (b) Corridor of length $N = 20$ (c) Corridor of length $N = 50$

(d) Taxi (e) Corridor (f) Maze

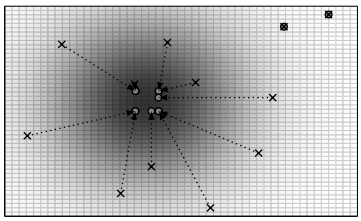
Figure 4: Performance of the P-ITERATION algorithm in terms of execution time. Results averaged over 30 runs. Bars represent the standard error. 4a to 4d plot the execution time as a function of the number of parameters in the corridor and taxi scenarios. 4e plots the execution time as a function of the corridor width and a single parameter. 4f plots the execution time as a function of the problem size of the maze scenario.

Table 1: Performance of P-ITERATION in terms of quality of the solutions. Results rounded to 2 decimal places. Scenarios where the optimal solution was found in more than 50% of the runs are in bold.

Scenario	N	$J(\theta_0)$	K	$F(\theta)$
Corridor	2	-1.40	1	**-1.23**
	5	-3.49	1	**-2.32**
	10	-5.61	1	**-3.86**
			2	**-3.86**
			3	**-3.86**
			4	**-3.87**
			5	-3.90
	20	-7.54	1	**-5.85**
			3	**-5.85**
			5	-5.87
			7	-5.89
			10	-5.92
	50	-9.00	1	**-8.12**
			5	-8.13
			10	-8.15
			15	-8.17
			25	-8.23
Maze	2	-1.40	1	**-1.23**
	6	-7.28	5	**-3.98**
	7	-7.97	6	**-4.51**
	11	-9.17	10	-6.16
	15	-9.56	14	-7.24
Taxi	—	80.13	1	**82.66**
			2	**83.43**
			3	**86.57**
			4	**87.04**
			5	**89.88**



(a) Performance in random scenarios



(b) Performance in water pouring scenario

Figure 5: 5a depicts the improvements in value J in the random scenarios, as a function of the problem size and number of random configuration parameters (K). Darker is better. The differences are significant, since due to the reward function and discount factor used the maximum $J(\theta)$ possible in this domain is 30. 5b depicts the radial kernel used and the solutions achieved when using the cost function with $\beta = 10$.

Figures 4a to 4d plot the execution time of P-ITERATION as the number of parameters increases on the corridor and taxi scenarios. In these four plots the execution time seems to degrade gracefully as the number of parameters increases. The two remaining Figures, 4e and 4f, plot the execution time as a function of the problem size ($\mathcal{X} \times \mathcal{A} \times \mathcal{X} \times K$). In Figure 4e the number of parameters is fixed to one, and only the state-space changes. In Figure 4f, both the state-space and number of parameters is growing, since in the maze scenario the number of parameters is linearly proportional to the state-space size. As such, these execution times seem to match our expectations of a quadratic and cubic growth, respectively.

Table 1 suggests the algorithm scales satisfactorily in terms of the quality of the solutions achieved. In all three scenarios P-ITERATION is able to find a configuration of the world θ such that $F(\theta) > J(\theta_0)$. In some cases the algorithm is actually able to consistently compute the optimal solution (in boldface). As discussed before, however, due to the non-convex nature of (4) as the problem size and the number of parameters increases the more likely it is to converge to local maxima.

5.2 Scenarios with complex parameterizations

To assess the applicability of our approach in a broader category of problems, we consider two more domains that require complex parameterizations of the transition probabilities.

First, we consider a set of random scenarios that can be seen as abstract environments represented as MDPs with varying dimensions. The transition probabilities are constructed randomly, following a method similar to that used in Garnet problems [1]. Similarly, we randomly construct a sparse immediate reward function, by assigning each state-action pair with a reward of 3 with a 25% chance.

We randomly select state-action pairs to which we associate a total of K configuration parameters. In particular, if (x, a) is one such state-action pair, a random subset of states $\mathcal{X}_{x,a}$ is randomly selected, and we associate a configuration parameter $\theta_{x,a,y}$ to the triplet (x, a, y), where

$$P_\theta(y \mid x, a) = \xi_{x,a} \frac{e^{\theta_{x,a,y}}}{\sum_{y' \in \mathcal{X}_{x,a}} e^{\theta_{x,a,y'}}} \tag{8}$$

with

$$\xi_{x,a} = 1 - \sum_{y \notin \mathcal{X}_{x,a}} P_\theta(y \mid x, a). \tag{9}$$

We also take $C(\theta) \equiv 0$ and μ_0 as the uniform distribution.

Figure 5a depicts the performance of P-ITERATION in these random scenarios. We ran experiments for different combinations of number of states, actions and configuration parameters. For each combination we generated 10 different random scenarios that were solved using 10 different seeds and 50 random restarts. The grayscale grids depict the difference $J(\theta) - J(\theta_0)$, with θ as the configuration returned by P-ITERATION. Results were averaged over the 10 runs of the 10 scenarios generated. We conclude that P-ITERATION is able to find configurations that lead to better policies, even in a highly unstructured scenario with a complex parametrization of the transition probabilities.

The second scenario considered models the decision process of a robotic agent that was trained to pour water in cups located at a specific position $\bar{\boldsymbol{\theta}}$. The state space $\mathcal{X} = \{I, S, F, A\}$ includes initial, success, failure and absorbing states. The action space $\mathcal{A} = \{q, x\}$ includes execution and "quitting" actions. The latter signals the robot is not confident in executing the task for the current configuration of the world $\boldsymbol{\theta}$—the position of the cup. The reward function penalizes the agent with -5 for executing q in state I. In that case, the agent also transitions immediately to state A. An execution x is penalized with reward -1 and has two possible outcomes. The agent transitions to S if it succeeds the task, and to F otherwise. In both outcomes, the agent ultimately transitions to state A, but when transitioning from F receives a penalty of -9.

The transition probabilities of the two execution outcomes were estimated using 6 pouring trajectories collected on a real robot. The probability of success of an execution was modeled as a radial kernel centered in $\bar{\boldsymbol{\theta}}$, the average final position of the trajectories collected. This simplifying assumption is often used in practice [15]. Formally,

$$P_\theta(S \mid I, x) = e^{-\xi \|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|_2^2}$$
$$P_\theta(F \mid I, x) = 1 - P_\theta(S \mid I, x),$$

where ξ is the width of the kernel, and was estimated empirically. We adopted $\mu_0(x) = \mathbb{I}(x = I)$ and a cost function $C(\boldsymbol{\theta}) = \beta \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_1$, for constant β and initial cup position $\boldsymbol{\theta}_0$. We ran experiments starting from 12 different configurations $\boldsymbol{\theta}_0$, with the average $J(\boldsymbol{\theta}_0)$ being approximately -4.57. For β with values 0, 5 and 10, P-ITERATION computed solutions $\boldsymbol{\theta}$ where the average $J(\boldsymbol{\theta})$ values were -1, -2.55, and -3.73, respectively. Figure 5b depicts the solutions achieved for $\beta = 10$. Note that, for this large β, there are two cases where it would be too costly to get a new configuration of the world where it is worth trying to execute the pouring action. As a result, the P-ITERATION algorithm determined that it is better to not change the world in those two cases.

5.3 Discussion

The experimental evaluation performed demonstrates the efficacy and applicability of both the P-ITERATION algorithm and our general problem formulation. The results show that our approach enables an agent to find new configurations of the world that allow for better policies, while accounting for the potential cost associated with shifting the world to such configurations. This is the case, even in domains with complex parameterizations of the transition probabilities and arbitrary cost functions, like the random and water pouring scenarios.

This sets our work apart from other ideas in the MDP literature. Namely, MDPs with imprecisely known parameters (MDPIPs), which may find new policies by making assumptions regarding some uncertainty (often modeled as a polytope) in the transition probabilities [3, 20, 5, 21]. MDPIPs are only able to model a restricted subset of the class or problems we

consider. Specifically, those problems where there is no cost for changing the world, *i.e.*, $C(\theta) \equiv 0$. Additionally, as far as we know, there is no MDPIP model capable of modeling complex feasibility constraints for the transition functions that span across multiple state action pairs, like those we provided on the corridor scenario, where the openness of a door affects transition probabilities associated with two different actions (Up, Down) and initial states.

Our work also differs from the approach of modeling possible changes to the world by directly including additional "world-changing" actions in the agent's set of actions $\mathcal{A}$. Note that possible changes to the world are often beyond the direct capabilities of the agent—for example, in the aforementioned corridor scenario the robot is not able to directly open the door. Consequently, from a conceptual point of view, this alternative approach suggests adding actions to $\mathcal{A}$ that the agent cannot directly perform, and that are not necessary to solve the original planning problem in the first place. Moreover, including these additional actions comes with significant computational disadvantages. In order for a change in the world to be persistent throughout the decision process, the state space must include information about it. This leads to an exponential growth of the state space as a function of the number of changes to the world allowed. Moreover, if continuous changes to the world are to be considered, a continuous state and action space MDP formulation becomes necessary. Continuous state and action space MDP formulations are well known to be much harder to solve [2].

6 Conclusion

This work is motivated by the fact that planning algorithms assume a given fixed world and a reward function, and then determine a policy that maximizes such reward in the given world. We consider the problem of reasoning at planning time about the counterfactual "what if the world were different"? Even assuming a cost to change the world, is there a world configuration that leads to a more rewarding policy?

We contribute a novel formulation that models this counterfactual, allowing an agent to reason over possible world configurations, with the goal of finding new and more rewarding optimal policies, while taking into account the underlying cost of changing the world. We showed that, in the case of scenarios described as Markov decision processes, this problem can be modeled as a constrained optimization problem, in which the agent balances the benefits arising from changing the world with the potential costs incurred in such change. We then presented a gradient-based algorithm for effectively solving this optimization problem.

The experimental evaluation performed shows the applicability of our approach on several scenarios of practical interest. In the scenarios tested, our approach was able to find new feasible configurations of the world that are associated with optimal policies more rewarding than those possible in the original world configuration. This was the case even in scenarios with complex parameterizations of the transition probabilities and arbitrary underlying cost functions for changes to the world.

References

[1] T.W. Archibald, K. McKinnon, and L.C. Thomas. On the generation of markov decision processes. *Journal of the Operational Research Society*, 46(3):354–361, 1995.

[2] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.

[3] K. V. Delgado, S. Sanner, and L. Nunes De Barros. Efficient solutions to factored MDPs with imprecise transition probabilities. *Artificial Intelligence*, 175(9-10):1498–1527, 2011.

[4] T. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligent Research*, 13(1):227–303, November 2000.

[5] R. Givan, S. Leach, and T. Dean. Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122(1-2):71–109, 2000.

[6] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel. Coming Up with Good Excuses: What to Do when No Plan Can Be Found. In *Proc. 20th Int. Conf. Automated Planning and Scheduling*, ICAPS'10, pages 81–88, 2010.

[7] K. Hauser. Minimum Constraint Displacement Motion Planning. In *Robotics: Science and Systems*, 2013.

[8] K. Hauser. The minimum constraint removal problem with three robotics applications. *Int. Journal of Robotics Research*, 33(1):5–17, 2014.

[9] K. Kim, G. Fainekos, and S. Sankaranarayanan. On the minimal revision problem of specification automata. *Int. Journal of Robotics Research*, 34(12):1515–1535, 2015.

[10] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] S. Klee, B. Ferreira, R. Silva, J. Costeira, F.S. Melo, and M. Veloso. Personalized assistance for dressing users. In *Proc. 7th Int. Conf. Social Robots*, pages 359–369, 2015.

[12] R. Knepper, S. Tellex, A. Li, N. Roy, and D. Rus. Recovering from failure by asking for help. *Autonomous Robots*, 39(3):347–362, 2015.

[13] G. Neu and C. Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proc. 23rd Conf. Uncertainty in Artificial Intelligence*, pages 295–302, 2007.

[14] M. Nicolescu and M. Mataric. Learning and interacting in human-robot domains. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(5):419–430, 2001.

[15] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013.

[16] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[17] S. Rosenthal, J. Biswas, and M. Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *Proc. 9th Int. Joint Conf. Autonomous Agents and Multiagent Systems*, pages 915–922, 2010.

[18] R. Silva, M. Faria, F.S. Melo, and M. Veloso. Adaptive Indirect Control through Communication in Collaborative Human-Robot Interaction. In *Proc. 2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2017.

[19] R. Silva, F.S. Melo, and M. Veloso. Adaptive symbiotic collaboration for targeted complex manipulation tasks. In *Proc. 22nd Eur. Conf. Artificial Intelligence*, pages 863–870, 2016.

[20] F. W. Trevizan, F. G. Cozman, and L. Nunes De Barros. Planning under risk and knightian uncertainty. In *Proc. 20th Int. Joint Conf. Artifical Intelligence*, IJCAI'07, pages 2023–2028, 2007.

[21] C.C. White and H.K. Eldeib. Markov decision processes with imprecise transition probabilities. *Operations Research*, 42(4):739–749, 1994.