Risk-aware Path Planning Using Hirerachical Constrained Markov Decision Processes

Seyedshams Feyzabadi and Stefano Carpin

Abstract— **Next generation industrial plants will feature mobile robots (e.g., autonomous forklifts) moving side by side with humans. In these scenarios, robots must not only maximize efficiency, but must also mitigate risks. In this paper we study the problem of risk-aware path planning, i.e., the problem of computing shortest paths in stochastic environments while ensuring that average risk is bounded. Our method is based on the framework of constrained Markov Decision Processes (CMDP). To counterbalance the intrinsic computational complexity of CMDPs, we propose a hierarchical method that is suboptimal but obtains significant speedups. Simulation results in factory-like environments illustrate how the hierarchical method compares with the non hierarchical one.**

I. Introduction

Motion planning is a fundamental problem for robots moving in a variety of environments, including industrial and manufacturing plants where robots and humans work side by side. Due to the intrinsic computational complexity of motion planning [5], a significant fraction of research has aimed just at determining whether a path exists, without attempting to optimize additional criteria. Recently, methods aiming at computing paths optimizing a given cost function have been developed, e.g., RRT* [14]. A typical cost function is path length, or energy consumption, but in practical scenarios there is typically more than one objective. Starting from a generic definition of risk, our objective is to compute paths that simultaneously try to minimize one objective function (e.g., path length) and contain one or more additional measures (e.g., risks). One possible approach is to combine these multiple metrics into a single objective function and then apply one of the existing algorithms, like RRT*. However, combining together heterogeneous quantities like path length and risks hinges on the introduction of conversion factors that are not necessarily straightforward to determine. In situations like these it is more intuitive to optimize with respect to one objective function (say path length), while imposing constraints on the others. Considering the inevitable disturbances affecting robot actions, this standpoint directly leads to the formalism of Constrained Markov Decision Processes (CMDP). While this topic is well studied and understood, its natural limitation is found in its computational complexity because even relatively simple problem instances can generate linear programs with tens of thousands of variables. For this reason, this method is not appealing when recomputation may be frequently needed. In this paper we experimentally explore the use of a hierarchical CMDP model to tackle this problem. As in many related hierarchical approaches, in order to gain in computational efficiency it will be necessary to accept solutions that will be in general suboptimal. The speedup however is remarkable and allows to frequent updates of the computed path.

Our method is parametric with respect to the definition of one or more risk functions defined as a function of the environment and the action taken by the robot. For example, if the robot is at a given location and is facing an obstacle, moving towards the obstacle itself may result in a collision. This specific combination of location and action can then be associated with a *risk* value assessing the potential for a detrimental outcome of the state/action pair (i.e., a collision). On the contrary, if starting from the same location the robot backs away from the obstacle, the associated risk would be lower. Similarly, if the robot is operating in a wide open area, the collision risk associated with basic maneuvers can be considered negligible. These examples motivate our assumption that risk is a function of state and action.

The remaining of the paper is organized as follows. In Section II we discuss related work. Section III summarizes basic facts in the area of MDPs and CMDPs. The hierarchical algorithm is presented in section IV. Simulations evidencing strengths and weaknesses of the proposed method are discussed in section V, whereas conclusions and directions for future work are presented in section VI.

II. Related Work

Robot motion planning has been extensively studied in the last decades and related literature is vast [6], [15]. In motion planning it has been customary to assume that the state (configuration) of the robot is known. Due to the inherent complexity of the problem, most efforts have been directed towards the development of algorithms capable of efficiently determining if a solution exists, whereas research where risk is taken into account has been much more limited. A recent paper by Sukhatme and collaborators [8] considers the problem of risk-aware motion planning for autonomous underwater vehicles. Two algorithms are presented. The first considers minimizing the expected risk of a plan and performs an graph search over a weighted graph where edges are labeled with expected risk. The second algorithm uses Markov Decision Processes to account for uncertainty in

S. Feyzabadi and S. Carpin are with the School of Engineering, University of California, Merced, CA, USA.

This work is supported by the National Institute of Standards and Technology under cooperative agreement 70NANB12H143. Any opinions, findings, and conclusions or recommendations expressed in these materials are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the funding agencies of the U.S. Government.

the motion model. Other works in the area of risk aware motion planning and hazard avoidance include [10] and and [16] where classic informed search techniques and genetic algorithms are used.

Literature in the area of hierarchical MDPs is rich too. One of the seminal approaches was presented in [17] where a hierarchical decomposition was proposed with the objective of studying a decision making problem at different time scales. Hierarchical methods have often been proposed with the objective of computing solutions to small subproblems that could be then reused as building blocks for solutions to larger problems. Hauskrecht et al., instead, embrace a standpoint similar to ours, i.e., they pursue a hierarchical approach with the objective of compressing the state space and then decreasing the time spent to compute a solution [13]. Other solutions were proposed in [3], [7], [12].

Constrained Markov Decision Processes have been used in numerous engineering domains [2], but their application in the area of robotics and automation is limited [9]. To the best of our knowledge, the use of hierarchical methods based on CMDPs for risk aware path planning appears to be an unexplored area, and in general hierarchical CMDPs have not been extensively investigated.

III. Constrained Markov Decision Processes

In this section we briefly summarize basic results and notation regarding Constrained Markov Decision Processes.

A. Total Cost Markov Decision Processes

Markov Decision Processes (MDPs) are used to solve decision making problems where the state is observable and the outcome of actions is stochastic [4]. We consider the simplest case of finite MDPs evolving in discrete time. A finite, stationary MDP is defined by a quadruple X, U, P, c:

- X is a finite state space with n elements. The state at time t is indicated by the random variable X_t.
- U is a collection of n finite sets, one for each state in X. $U(x)$ is the set of actions that can be applied in state x. A_t is the action taken at time t.
- P is the transition probability function. We define $P^a_{xy} = P(X_{t+1} = y | X_t = x, A_t = a)$ as the probability that the state transitions from x to y when action a is taken. This probability is assumed to be stationary.
- $c : X \times U \to \mathbb{R}_{\geq 0}$ is a cost function. $c(x, a)$ is the cost incurred when applying action a while in state x.

A policy π is a rule defining the action to be taken at time t. Given the stochastic nature of state transitions, for a given initial state x_0, a policy π induces a discrete time stochastic process $\zeta = (X_t, A_t)$, where X_t is the random variable for the state at time t and A_t is the random variable for the action at time t. Through the function c, different costs can be associated to every realization of the stochastic process ζ. Common choices include 1) finite horizon total cost; 2) infinite horizon discounted cost; 3) infinite horizon average cost; and 4) infinite horizon total cost. For the applications considered in this paper, infinite horizon total cost are the most relevant, so in the following we exclusively limit our discussion to this case. For the definition of total cost, it is necessary to assume that the MDP is transient and make some assumptions about the costs, otherwise the total cost defined in the following may not exist. An MDP is transient if the state space X can be partitioned into two sets X' and M such that for every policy π:

1) $\sum_t P^\pi(X_t = x) < \infty$ for every $x \in X'$;
2) $P^a_{xy} = 0$ for each $x \in M$ and $y \in X'$,

where $P^\pi(X_t = x)$ is the probability that $X_t = x$ when following policy π. We assume $c(x, a) = 0$ for each $x \in M, a \in U(x)$. Under these conditions the total cost of π is

$$c(\pi) = \sum_t E_\pi c(X_t, A_t)$$

where E_π is the expectation induced by the policy π. Note that, because we assumed that the MDP is transient, the above cost is well defined and is finite. For a given transient MDP, the optimal policy is therefore defined as follows:

$$\pi^* = \arg\min_\pi c(\pi).$$

Total cost MDPs are relevant for engineering applications because they model systems where the time taken to complete a given task (modeled by the event $X_t \in X$) is not known upfront, but is known to be finite. This is in contrast to the case of finite horizon total cost, where one has to assume the completion time is known beforehand, and infinite horizon discounted cost, where one assumes the process evolves indefinitely. In a path planning scenario, one is interested in designing motion policies ensuring the goal location is reached in finite time, but due to the stochasticity in the motion model the precise time cannot be anticipated. Then, infinite horizon total cost is the most appropriate model.

It is well known that for MDPs there always exists an optimal, stationary, deterministic, Markov policy. Therefore the optimal policy can be described by a function $\pi : X \to U$ mapping states into action. π is deterministic because it associates just one action to every state, it is stationary because it does not depend on time, and it is Markov because it depends only on the current state and not on the previous history. Note also that the policy depends on the partition X', M, but is independent from the initial state X_0. Various methods have been proposed to compute optimal policies, including formulations based on linear programming, and iterative algorithms like value iteration and policy iteration. From a practical point of view, policy iteration and value iteration are the algorithms of choice when solving MDPs.

B. Total Cost Constrained Markov Decision Processes

Total costs Constrained Markov Decision Processes (CMDP) extend the MDP model by including additional costs subject to constraints. A CMPD is defined by X, U, P, c, d_i, D_i where X, U, P, c are defined as for the MDP case and

- $d_i : X \times U \to \mathbb{R}_{\geq 0}$, $1 \leq i \leq k$ are k additional cost functions. $d_i(x, a)$ is the ith additional cost incurred when applying action a in state x.

- $D_i \geq 0$ are k positive constants to constrain the additional costs induced by the functions d_i and defined in the following.

Despite the similarity between the definition of MDP and CMDP some notable differences exist (the reader is referred to [1] for a comprehensive discussion). The two main ones are the following. First, the optimal policy (irrespectively of the cost model), may in general require randomization (as opposed to the deterministic case for MDPs). Second, the optimal policy is not independent from the initial state and it in general depends on the mass distribution defining the initial state. The initial mass distribution will be indicated by the function $\beta(x) = P(X_0 = x)$. A transient total cost CMDP is defined like the transient MDP, but the first condition is substituted by the following

$$\sum_t P_\beta^\pi(X_t = x) < \infty$$

where $P_\beta^\pi(X_t = x)$ indicates the probability that state X_t is x when following policy π and under the assumption that the initial state X_0 is distributed according to β. In order to define the total cost, we furthermore need to assume $d_i(x, a) = 0$ for each x, a and each d_i. Total costs can then be defined as follows (note the dependency on both the policy π and the initial distribution β)

$$c(\pi, \beta) = \sum_t E_{\pi,\beta} c(X_t, A_t)$$

$$d_i(\pi, \beta) = \sum_t E_{\pi,\beta} c(X_t, A_t).$$

Solving a CMDP means determining the optimal policy for the following constrained optimization problem:

Optimization problem CMDPOPT

$\pi^* = \arg\min c(\pi, \beta)$

s.t. $d_i(\pi, \beta) \leq D_i,\ 1 \leq i \leq k.$

Another major difference between MDPs and CMDPs is found in the solving algorithms. CMDPs cannot be solved using iterative algorithms like value iteration or policy iteration. Instead, the method of choice is solving a linear program defined as follows. Let $\mathcal{K} = \{(x, a), x \in X', a \in U(x)\}$ and let $\rho(x, a)$ be $|\mathcal{K}|$ optimization variables associated to each element in $\mathcal{K}$. A fundamental theorem in the theory of CMDPs (see [1]) establishes that the optimization problem CMDPOPT has a solution if and only if the following linear program is feasible:

$$\begin{aligned} \min_\rho \ & \sum_{(x,a)\in\mathcal{K}} \rho(x, a) c(x, a) \qquad (1) \\ \text{s.t.} \ & \sum_{(x,a)\in\mathcal{K}} \rho(x, a) d_i(x, a) \leq D_i \quad 1 \leq i \leq k \\ & \sum_{(y,a)\in\mathcal{K}} \rho(y, a)(\delta_x(y) - \mathcal{P}_{yx}^a) = \beta(x) \ \forall x \in X' \\ & \rho(x, a) \geq 0 \quad \forall (x, a) \in \mathcal{K}. \end{aligned}$$

Moreover, the optimal solution to the linear program induces an optimal stationary policy for the CMDP. The optimal policy is defined as

$$\pi^*(x, a) = \frac{\rho(x, a)}{\sum_{a\in A(x)} \rho(x, a)} \quad x \in X', a \in U(x), \qquad (2)$$

where $\pi^*(x, a)$ is the probability of taking action a when in state x. The policy can be arbitrarily defined when $\sum_{a\in A(x)} \rho(x, a) = 0$. Note that this policy is stationary and Markov (depends only on x and not on past history), but it is not deterministic, as evident from its definition. We conclude this short recap on CMDPs remarking that the constraint on d_i are satisfied in expectation, i.e., not every realization of ζ will satisfy the constraint, but the constraint will be satisfied in expectation.

C. Connection to risk aware path planning

As discussed in Section II, MDPs have been used for risk aware motion planning in the past and recently, e.g., in [8]. However, relying only on MDPs is limiting for the following reasons. First, in practical applications one is often interested in performance measures combining multiple criteria, like e.g., finding the shortest path subject to a bound on the risk, or, alternatively, finding the path with the lowest risk subject to a bound on the length. These practically relevant objectives cannot be accommodated using MDPs only, but can be naturally formulated as CMDPs problems. Moreover, the CMDP framework allows to consider an arbitrary number of additional constraints. One can then for example seek the shortest path subject to different bounds to different types of risk. In addition, both c and d_i are defined as functions of state and action. This allows to model risks that are function of the state only, or of the state and action. For example, in an industrial scenario where autonomous forklifts move around in an environment shared with humans, one can associate higher risk to areas where human presence is frequent. In this case, risk would be a function of the state only, assuming that the state includes the location of the forklift. On the other hand, one could also define risk functions that are function of both state and action. For example, one could define a risk function d_i depending on both components of (x, a). For the same state x, such function could assign a high risk to an action moving the robot towards an obstacle, and a lower risk to a different action moving away from the obstacle. In this way one can manage problems similar to the region of inevitable collision [11].

IV. Planning with hierarchical CMDPs

One of the well known limitations of MDPs and CMDPs relies on their computational complexity. CMDPs, in particular, require the solution of linear program whose size may quickly become unmanageable (see section V for some numbers). For this reason, hierarchical approaches can be a viable alternative, as long as there is a clear understanding that this will lead to suboptimal solutions. Hierarchical methods in the area of MDPs have been practiced in the

past mainly with the objective of determining policies that can be reused, see e.g., [13]. We instead pursue hierarchical solutions with the objective of reducing the computational time. In particular, we are interested in reducing the computational effort with the eventual objective of enabling reactive replanning when changes in the environment are detected by the robot. Changes may for example concern a rearrangement of the obstacles on the floor plan, or a person moving from one area to another, with the consequent necessity to update the associated risk.

Our approach is based on the idea of aggregate states (see [4], Vol 1, pg. 321), an idea that has been proposed for MDPs, but needs to be adapted for CMDPs. In the following we make the assumption that the action set $U(x)$ is identical for all states. This is consistent with the application scenario we present in the following and can also be imposed in the general case.Without lack of generality, in the interest of simplicity we furthermore assume that $\beta(x) = 1$ for just one state and is 0 for all other states, and we assume there is just one goal state. These two states will be indicated as S (start) and G (goal).

The general idea is as follows. We start from a given CMDP with a large state space and build a hierarchical version through state aggregation. The aggregate CMDP is then solved and a policy over the aggregate states is determined. In order to map the high level policy back into actions that can be executed in the original CMDP, a smaller CMDP is solved for each aggregate state traversed by the high level policy. Since action execution is stochastic, one cannot anticipate the precise sequence of aggregate states that will be traversed when the policy is followed. Therefore the solution of the smaller CMDPs is interleaved with execution, because the precise sequence of traversed aggregate states cannot be predicted upfront (see Figure 1).

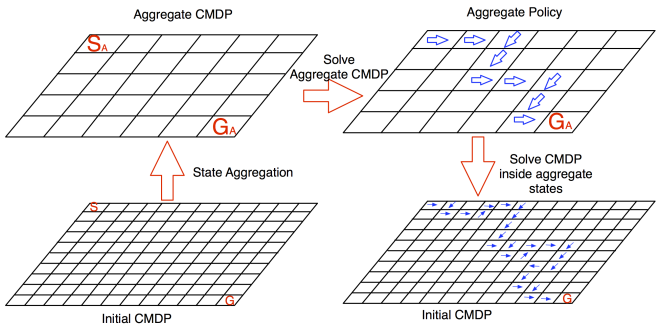


Fig. 1: Large CMDPs can be solved through aggregation. Note that for CMDPs both the start and the goal state (S and G in the figure) need to be specified, as opposed to the MDP case where an optimal policy is specific by the goal state only.

In the following, quantities with the subscript A refer to the aggregate CMDP. First, to define the state space X_A of the aggregate CMDP we partition the state space X into a set of m aggregate states $A_1, A_2, \ldots, A_m$. Each aggregate state represents a set of states in the original CMDP. The problem of how to split X into X_A is a long standing issue. A rule of thumb is to group together states with similar costs, when this is possible. In our application, targeting path planning in planar environments this calls to grouping together states that are nearby in a metric sense. Indeed, nearby states have usually comparable distances to the goal location. Moreover, it is reasonable to assume that risks have a local dependency on the state, so nearby states (or state/action pairs) have comparable risks. The problem of how to partition X using different criteria is left for future investigation. Two quantities called aggregation and disaggregation probabilities are defined to represent the aggregation. The aggregation probability $w_{ij} = 1$ if $x_j \in A_t$ and 0 otherwise. The disaggregation probability is $q_{si} = \frac{1}{|A_s|}$ if $x_i \in A_s$ and 0 otherwise. These definitions correspond to the so-called *hard assignment*, i.e., every state in the original CMDP is assigned to one and only one aggregate state. *Soft* assignments are possible too, with states possibly belonging to multiple aggregate states, but this extension is left to future investigation. The start and goal states S, G in the original CMDP obviously induce start and goal states S_A and G_A in the aggregate CMDP because S and G belong to one and only aggregate state.

Next, it is necessary to define the action set, transition probabilities, and the costs c_A and $d_{i,A}$ for the aggregate CMDP. Since we assumed all states in the original CMDP had the same action set U, it follows that we can assume all states in the aggregate CMDP also have the same action set, and this is also U. Every action executed in a certain aggregate state A_i has an intended effect in terms of state transition (e.g., this can be the state with towards which there is the highest transition probability). The function NextState(A_t, u), used in the implementation of algorithm 1 returns such state. Moreover, we define the frontier between macrostate A_t and A_n as the set of states in A_n that can be reached in one transition from one state in A_t. Transition probabilities between state A_s and A_t in the aggregate CMDP are then defined as follows:

$$P_{A_s A_t}(u) = \frac{1}{|A_s|} \sum_{i \in A_s} \sum_{j \in A_t} P_{ij}(u).$$

Similar definitions can be given for the costs:

$$c_A(A_s, u) = \frac{1}{|A_s|} \sum_{i \in A_s} c(x_i, u)$$

$$d_{i,A}(A_s, u) = \frac{1}{|A_s|} \sum_{i \in A_s} d_i(x_i, u).$$

Algorithm 1 presents an algorithmic sketch of the idea we described. The algorithm starts creating and solving the aggregate CMDP (line 2 and 3). Then, it extracts the optimal policy for the aggregate CMDP and starts to follow it (while loop). Inside the loop, the CMDPs for the aggregate states are solved *on demand*, i.e., the CMDP for an aggregate state A_i is created and solved only when the state enters the corresponding aggregate state. Note also that, because of the

Data: CMDP $= (X, U, P, c, d_i, D_i, S, G)$
Build CMDP $(X_A, U_A, P_a, c_a, d_{i,A}, D_{i,A}, S_A, G_A)$;
Solve aggregate CMDP (Opt probl. 1);
Extract optimal aggregate policy π_A^* (Eq. 2);
$x \leftarrow S$;
while $x \neq G$ **do**
  Determine state A_t for which $w_{xt} = 1$;
  if $A_t = A_G$ **then**
    $GoalSet \leftarrow \{G\}$
  end
  else
    $u \leftarrow \pi_A^*(A_t)$;
    $A_n \leftarrow$ NextState(A_t, u);
    $GoalSet \leftarrow$ Frontier(A_t, A_n)
  end
  $\pi_L \leftarrow$SolveLocalCMDP$(x, GoalSet)$;
  repeat
    Follow policy π_L and update x
  until x *reaches GoalSet or exits* A_t;
end

Algorithm 1: Algorithmic Sketch

unpredictability in the motion of the robot, it is possible that when the robot tries moving from macro state A_i to A_j it instead reaches a different macro state A_k (think for example to the case of a robot moving along the boundary of two macro states). This event does not create a problem. A policy over the macro states is preliminarily computed, so even if the state deviates from the intended trajectory the corresponding high-level policy is always available. Next, CMDPs for the macro states are solved on the fly, so an unpredicted transition into a different macro state can be dealt with by the algorithm. Note also that the algorithm does not solve CMDPs for the macro states not visited during the main while loop. The interleaved planning and execution strategy is essential to limit the computational cost.

V. Experimental Evaluation

A. Setup

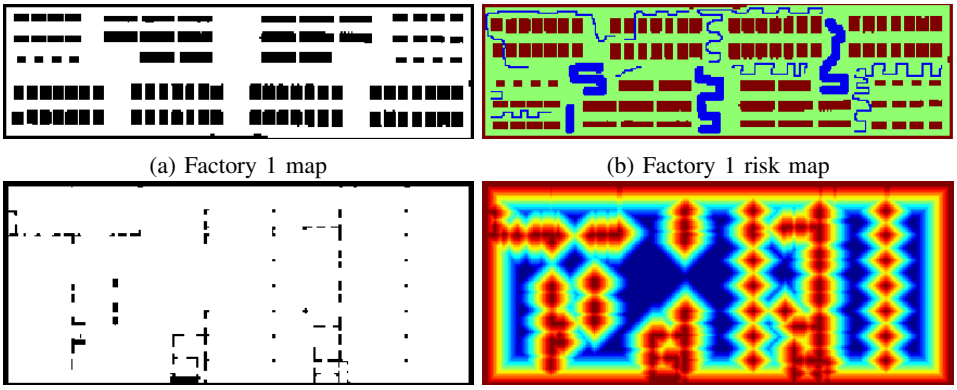
In this section we present some experiments outlining the advantages and limitations of the hierarchical decomposition. Figure 2 shows the floorplans of tho factories retrieved from the web, with white pixels encoding free space and black pixels indicating obstacles. The same picture shows two corresponding risk maps. For sake of simplicity and ease of visualization, we assumed that risk is a function of the state only, and risk is defined as distance from the closest obstacle. We consider just one type of risk, but, as evident from the discussion in sections III and IV, more than one risk can be included. Both maps are divided into equally sized cells. The first maps consist of 78×272 cells, whereas the second one include 111×270 cells. We assume that the robot fits inside one of the cells, and furthermore assume 4-connectivity, i.e., from every cell the robot can move up, down, left, right, assuming the neighboring cell is not occupied by an obstacle. These numbers define the number of variables in the linear program given in Eq. 1 and shows why a hierarchical approach is necessary. For Factory 1 the linear program has 54542 variables, whereas for Factory 2 it has 114862 variables. Evidently, the time needed to setup and solve these large optimization problems prevents rapid replanning when needed, thus showing the necessity to go for a hierarchical approach. Risk is here defined as the sum of the risks accrued throughout the path, i.e., it is the sum of the risks of the traversed cells.

The stochastic motion model is defined as follows. When the robot executes action u from state x trying to reach state y (say moving up on the grid), it succeeds with probability 0.8 and fails with probability 0.2. When it fails, it may remain in x or move to any of the free cells adjacent to x and different from y (all with equal probability).

B. Numerical evaluation

We contrast the performance of the hierarchical approach with the non hierarchical one. In particular, we compare the time spent to compute a solution, the average path length and the average risk. In order to compare the time spent to compute a solution, for the non hierarchical approach we consider the time spent solving the linear program given in Eq. 1. For the hierarchical method we give the sum of the time spent solving all instances of linear programs Eq. 1 (see Algorithm 1). In both cases the linear program is solved using Matlab's builtin command to solve linear programs (`linprog` – in particular we use the interior-point method). We compare the time spent solving instances of the linear program because this is where most of the time is spent, and the remaining time (e.g., setting up the matrices for the linear program) strongly depends on the characteristics of the used programming language (Matlab, in this case). Moreover, with respect to timing, emphasis should be given to the ratios between hierarchical and non-hierarchical, and not to the absolute values. For each environment we consider five different couples of start/goal points and we test both the hierarchical and non-hierarchical methods on the same problem instances. For the case Factory 1, the hierarchical CMDP is obtained splitting the original grid into a coarser 6×8 grid with , whereas Factory 2 was divided into a 9×10 grid.

In both cases we use two different methods. In the first approach, the aim is to minimize risk while satisfying a constraint on path length. For these two particular maps, the minimum distance between the chosen start/goal point pairs is the Manhattan distance. We set the constraint on path length to be 40 % higher than the Manhattan distance. In the second approach we minimize path length while keeping a constraint on risk. This dual setup shows the flexibility of the CMDP approach. In certain situations it may be simpler to estimate and then bound one cost (e.g., path length) and optimize for the other (e.g., risk), whereas in other scenarios the roles of the costs may swap. Figure 3 shows two sample paths produced by these two approaches. The first picture shows the solution obtained while trying to minimize path



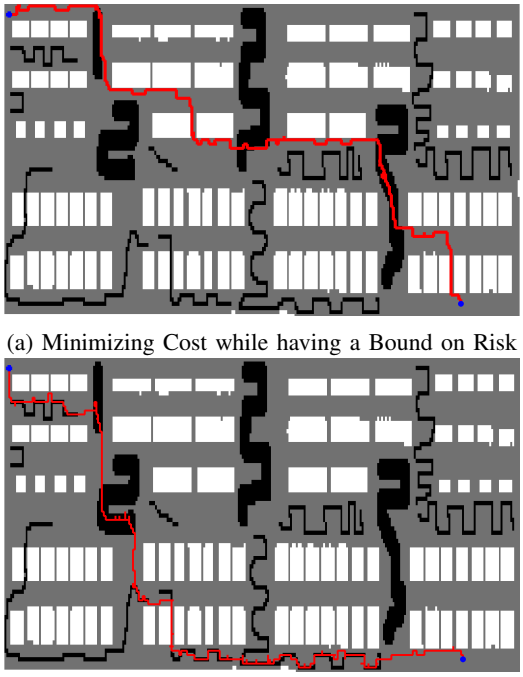
(a) Factory 1 map (b) Factory 1 risk map

(c) Factory 2 map (d) Factory 2 risk map

Fig. 2: The two test environment used for experimental evaluation. The left subfigures show the structure of the environment, whereas the right subfigures show the associated risk maps (warmer colors indicate riskier states).

length. The second instead shows a path to minimize risk. Table I and II show the performance of the non-hierarchical methods for the five test cases in the two environments ("MC" stands for "Minimizing Cost, Having a Bound on Risk". "MR" stands for "Minimizing Risk, Having a Bound on Cost") whereas table III and IV show the performance of the hierarchical methods for the five test cases in the two environments. Statistics for path length and risk are computed on the basis of 100 repeated trials.

This preliminary round of simulations, although limited, allows to draw some interesting insights. Comparing tables I with III and II with IV the advantage (and limitations) of the hierarchical approach emerges. The speedup, in terms of computational time, is large and varies from a factor of 15 to a factor of 500. When comparing path length between the hierarchical and the non-hierarchical solution, no substantial loss is observed. The situation less favorable when the risk metric is considered. While for the first environment results are satisfactory, the the second environment a noticeable performance deterioration is observed in six out of ten cases. We believe that this problem happens due to an improper splitting of the environment into macro-cells. Macro-cells need to have similar characteristics, otherwise choosing a high level path is not effective enough to help reduce risk on low level path. We guess creating macro-cells made of cells having similar risk values will improve this part of the algorithm.



(a) Minimizing Cost while having a Bound on Risk

(b) Minimizing Risk while having a Bound on Cost

Fig. 3: Two sample paths from the upper left corner to the lower right corner in the Factory 1 environment: Figure (a) shows the result of minimizing the path length while the bounding risk. Figure (b) shows the result of an attempt to minimize risk while bounding path length.

VI. Conclusions and Future Work

In this paper we have considered the problem of risk aware path planning. Risk aware path planning is a relevant problem when robots and humans operate in the same environment, for example in the envisioned next generation production plants, where numerous autonomous moving robots will share the same working environments with humans. Our tenet is that in this domain the use of Constrained Markov Decision Process offers an interesting, yet under explored tool to combine together multiple objectives, e.g., paths that are short and at the same time meet constraints on the acceptable risk. In order to counter the inherent complexity of CMDPs, we have studied a hierarchical method that produces suboptimal solutions, but exhibits a notable speedup while maintaining a comparable performance. There exist numerous areas for future work in this domain. First, we will study whether it is possible to derive analytic, informative bounds to anticipate the loss of performance in the hierarchical approach. While this problem is in general hard, our objective is to focus on specific instances of the problem (e.g., similar to those presented in Section V) to

Test Case	Time(s)		Avg. Length		Std Length		Avg. Risk		Std Risk	
	MC	MR	MC	MR	MC	MR	MC	MR	MC	MR
Case 1	53.2	1347.3	413.9	446.6	17.0	24.2	1682.8	1326.6	94.7	77.2
Case 2	75.3	1226.8	104.0	114.4	10.6	11.9	416.0	311.3	70.9	33.9
Case 3	72.0	58.3	104.9	114.2	8.8	10.6	486.4	351.0	43.5	35.1
Case 4	85.7	67.5	372.3	388.4	17.7	19.5	1462.2	1282.9	81.2	74.9
Case 5	49.5	1351.3	142.7	176.8	10.5	15.4	558.2	436.2	57.7	51.0

TABLE I: Performance of the non-hierarchical method on Factory 1.

Test Case	Time(s)		Avg. Length		Std Length		Avg. Risk		Std Risk	
	MC	MR	MC	MR	MC	MR	MC	MR	MC	MR
Case 1	307.0	258.8	487.1	508.7	22.6	24.2	2294.9	1760.3	362.5	129.4
Case 2	350.1	1343.2	361.9	414.7	17.5	23.6	1513.4	1132.1	176.0	105.2
Case 3	237.7	389.9	156.7	164.4	12.9	14.0	685.6	485.2	145.6	67.7
Case 4	303.0	195.7	172.8	180.9	16.4	14.4	273.9	230.6	49.5	33.9
Case 5	283.4	369.2	264.2	285.5	12.6	15.8	1406.4	959.6	210.1	99.3

TABLE II: Performance of the non-hierarchical method on Factory 2.

Test Case	Time(s)		Avg. Length		Std Length		Avg. Risk		Std Risk	
	MC	MR	MC	MR	MC	MR	MC	MR	MC	MR
Case 1	3.9	12.5	444.4	445.4	18.1	22.4	2134.9	1819.5	96.6	112.3
Case 2	2.0	2.6	108.7	113.5	9.9	11.2	465.0	405.6	71.8	60.7
Case 3	3.1	5.1	123.3	134.5	11.3	14.1	560.9	505.1	53.9	58.8
Case 4	7.9	7.1	395.2	389.9	20.6	17.3	1630.5	1435.8	91.8	91.6
Case 5	2.3	2.5	147.6	150.3	14.3	11.7	595.4	548.1	77.9	49.0

TABLE III: Performance of the hierarchical method on Factory 1.

Test Case	Time(s)		Avg. Length		Std Length		Avg. Risk		Std Risk	
	MC	MR	MC	MR	MC	MR	MC	MR	MC	MR
Case 1	8.2	12.2	495.6	508.3	21.2	24.3	6561.4	2908.1	581.2	283.4
Case 2	7.9	10.4	369.8	387.6	17.5	20.9	4606.8	1893.6	428.8	228.04
Case 3	9.9	7.7	165.0	167.4	12.8	14.6	901.7	1139.7	135.0	233.7
Case 4	6.6	8.1	174.5	186.7	11.3	17.2	495.1	274.4	58.6	44.4
Case 5	10.5	11.5	271.4	292.6	12.5	17.2	1969.9	1445.4	156.8	132.3

TABLE IV: Performance of the hierarchical method on Factory 2.

exploit the inherent structure of the problem to derive a bound. Another area for future investigation is to determine how to map in a principled way risk bounds between the flat and the hierarchical CMDPs. Moreover, we are exploring different methods to cluster states to mitigate some of the problems emerged in the simulated runs. In the long term, our goal is to test this framework on a real robotic platform.

REFERENCES

[1] E. Altman. Constrained Markov decision processes with total cost criteria: Occupation measures and primal LP. *Mathematical methods of operations research*, 43:45–72, 1996.

[2] E. Altman. *Constrained Markov Decision Processes*. Stochastic modeling. Chapman & Hall/CRC, 1999.

[3] J.L. Barry, L.P. Kaelbling, and T. Lozano-Pérez. Deth*: Approximate hierarchical solution of large markov decision processes. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[4] D. P. Bertsekas. *Dynamic Programming & Optimal Control*, volume 1 and 2. Athena Scientific, 2005.

[5] J. Canny. *The Complexity of Robot Motion Planning*. MIT Press, 1988.

[6] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of robot motion*. MIT Press, 2005.

[7] P. Dai, M.D.S. Weld, and J. Goldsmith. Topological value iteration algorithms. *Journal of Artificial Intelligence Research*, 42:181–209, 2011.

[8] A. A. de Menezes Pereira, J. Binney, G. A. Hollinger, and G S. Sukhatme. Risk-aware path planning for autonomous underwater vehicles using predictive ocean models. *Journal of Field Robotics*, 30(5):741–762, Oct 2013.

[9] X. Ding, A. Pinto, and A. Surana. Strategic Planning under Uncertainties via Constrained Markov Decision Processes. In *IEEE International Conference on Robotics and Automation*, pages 4568–4575. IEEE, 2013.

[10] L. De Filippis, G. Guglieri, and F. Quagliotti. A minimum risk approach for path planning on UAVs. *Journal of Intelligent Robotic Systems*, 61(203-219), 2011.

[11] T. Fraichard and H. Asama. Inevitable collision states – a step towards safer robots? *Advanced Robotics*, 18(10):875–884, 2004.

[12] C. Guestrin and G. Gordon. Distributed planning in hierarchical factored mdps. In *Proceedings of the Eighteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 197–206, 2002.

[13] M. Hauskrecht, N. Meuleau, L.P. Kaelbling, T. Dean, and C. Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the Fourteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 220–229, 1998.

[14] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotic Research*, 30(7):846–894, 2011.

[15] S.M. LaValle. *Planning algorithms*. Cambridge academic press, 2006.

[16] A.R. Soltani, H. Tawfik, J.Y. Goulermas, and T. Fernando. Path planning in construction sites: performance evaluation of the Dijskstra, A*, and GA search algorithms. *Advanced engineering informatics*, 16:291–303, 2002.

[17] R.S. Sutton. TD models: modeling the world at a mixture of time scales. In *Proceedings of the International Conference on Machine Learning*, pages 531–539, 1995.