

A Horizon-slicing Approach to Minimum Obstacle Displacement Planning for Robot Navigation

Antony Thomas^a, Giulio Ferro^b, Fulvio Mastrogiovanni^b, Michela Robba^b,
Marco Baglietto^b

^a*Robotics Research Center, IIIT Hyderabad, Hyderabad 500032, India.*

^b*Department of Informatics, Bioengineering, Robotics, and Systems Engineering,
University of Genoa, Via All'Opera Pia 13, 16145 Genoa, Italy.*

Abstract

In this paper, we investigate the *Minimum Obstacle Displacement Planning* problem from a robot motion planning perspective. The problem involves determining a feasible path to a goal location by displacing movable obstacles when no collision-free path initially exists. We show that this problem is computationally challenging and, in particular, NP-hard when obstacles are modeled as polygons in the plane. Besides an exact formulation of the minimum obstacle displacement problem generalizing other problems in the literature, and the associated optimal solution, this paper proposes an approximate solution that is *less intensive* from a computational standpoint, and differs from the optimal solution by a fraction of the optimal cost, being able to trade-off between path length and amount of obstacle displacements.

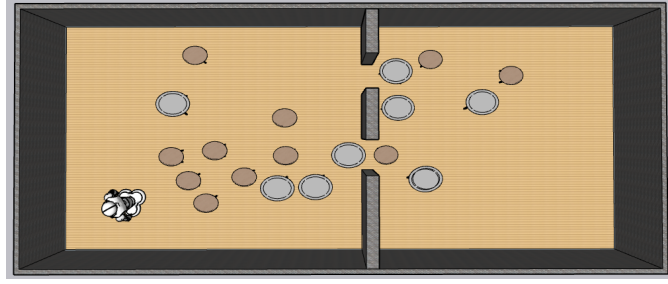
Keywords: Motion Planning, Movable Obstacles, Computational Complexity, Optimization

1. Introduction

Classical approaches to robot motion planning aim at finding a feasible path from a start to a goal location [1]. The feasibility of a motion plan is scenario-

Email addresses: antony.thomas@iiit.ac.in (Antony Thomas),
giulio.ferro@unige.it (Giulio Ferro), fulvio.mastrogiovanni@unige.it (Fulvio Mastrogiovanni), michela.robba@unige.it (Michela Robba), marco.baglietto@unige.it (Marco Baglietto)

specific. It involves different constraints, such as the robot’s mechanical design, its actuators’ performance, the robot’s perception capabilities, uncertainties in state estimation and control signals, and the employed collision avoidance strategy. In principle, a complete path planner systematically explores the search space induced by the environment and the available planning information, examining all candidate paths consistent with its search strategy. If no feasible path exists because of the workspace configuration or due to constraints violation, the planner terminates reporting failure, or otherwise planners complete in probability can be used [2]. However, in several application domains, it could be legitimate to evaluate the possibility of relaxing some of the constraints hindering a feasible path. For example, let us consider a robot arm tasked with reaching a target object in a cluttered tabletop setting. It may be the case that no collision-free path exists due to the specific arrangement of objects on the table. However, objects themselves could be moved aside, therefore allowing the robot to reach the target object. Likewise, mobile robots may reach goal locations provided that certain obstacles could be rearranged or moved aside. This may involve, for example, pushing aside chairs or other furniture, as well as opening doors along the path. The *Minimum Constraint Displacement* (MCD in short) problem introduced by Hauser [3] seeks to minimize the amount of *space* by which one or more objects may have to be displaced in order to yield a feasible path. The approach by Hauser treats objects in so far as they can be modeled as *constraints* for the motion planner. Therefore, in addition to relocating obstacles or modifying their configurations, the same framework demonstrates how a humanoid robot may modify its own body configuration to climb a ladder, for instance, by slightly increasing its hip angle limits, as well as finger and thumb strengths, that is, the mechanical constraints. All the constraints are represented in a graph, in particular as sub-graphs involving, for each obstacle, its candidate displacements (that is, a set of sampled displacements), which are associated with cost functions subject to a minimization procedure. One of the main contributions of [3] is the proof that the discrete version of MCD is NP-hard. Two approaches to solve the problem are discussed,



(a) A Pepper robot navigating amidst circular, movable objects.



(b) A mobile manipulator in a warehouse scenario.

Figure 1: (a) A top-down view of a possible scenario where a simulated Pepper humanoid robot is required to navigate to the right-hand side of the environment, and must displace obstacles to that aim. (b) A mobile manipulator, restricted to moving between the green lines, encounters an obstacle blocking its path. To proceed, the manipulator must push the obstacle aside before continuing along the designated trajectory. The warehouse environment is simulated in Gazebo by modifying an off-the-shelf model available at https://github.com/belal-ibrahim/dynamic_logistics_warehouse.

namely, a theoretically exact method, which has exponential time complexity in the worst case, and an approximate (greedy) approach providing *fast* solutions, but characterized by an unbounded approximation error.

This work focuses on scenarios in which a mobile robot must employ simple manipulation actions (such as, for example, pushing an object) when the goal location is not reachable due to the peculiar configuration of the environment. A

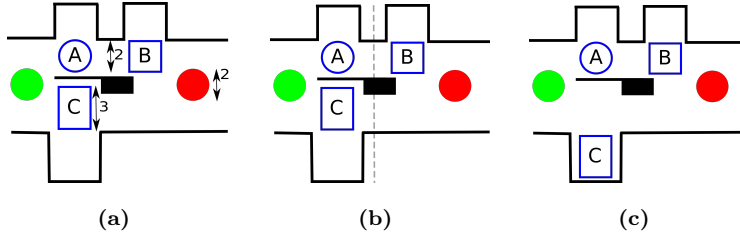


Figure 2: An example of the MOD problem not satisfying the optimal substructure property. (a) The robot must move from left (start location in green) to right (goal location in red) in the presence of three movable obstacles, referred to as A , B , and C . (b) A solution to the MOD problem splitting it into two sub-problems (left and right of the dashed line). This leads to a feasible path by moving obstacles A and B with a total displacement magnitude of 4 units. (c) The optimal solution is obtained by displacing obstacle C by 3 units.

first example is shown in Fig. 1a, where a Pepper robot must displace objects (tables and chairs) to navigate through the passages between two areas. A second example, illustrated in Fig. 1b, shows a mobile manipulator operating in an warehouse environment, which has to move obstacles out of its path, which is constrained by the green lines. Specifically, this paper is about how a robot can reach an assigned goal location by displacing obstacles while minimizing the total amount of obstacle displacements, and being able to trade off the number of obstacles to be displaced and the corresponding amount. In this work, we formulate a *Minimum Obstacle Displacement* (MOD) problem that minimizes a weighted cost of (a) total obstacle displacement magnitudes, and (b) robot path length, as the robot navigates to the goal location.

This problem is shown to be NP-hard in Section 3, and therefore it is appropriate to look into approximate methods. We introduce one such method, which we refer to as *Horizon Slicing for MOD* (HS4MOD in short), by dividing a MOD problem into (a sequence of) sub-problems. Solving MOD by separating it into sub-problems is an approximation as MOD does not exhibit the optimal substructure property [4], that is, optimal solutions to sub-problems – when combined – do not necessarily form an optimal solution to the overall problem.

This can be easily illustrated by an example. Let us consider the simple scenario shown in Fig. 2. The robot is required to move from a start (in green) to a goal (in red) location in the presence of movable obstacles (A , B , and C in the Figure). The robot can reach the goal location either by displacing obstacles A and B (that is, following the *upper* path) or by displacing obstacle C (the *lower* path). We note here that the start and goal locations are chosen for the sake of this example such that the robot path length for both the upper and the lower paths are equal. In Fig. 2b, the problem is split into two sub-problems. In the first sub-problem, the robot can either displace obstacle A by 2 space units or obstacle C by 3 units. Since minimum displacement is solicited (and path lengths are equal as argued above), the robot chooses to move obstacle A by 2 units. On the one hand, if the robot moved through the upper path by displacing A , in the next sub-problem it would have to move obstacle B by 2 units to reach the goal location, and therefore the total displacement magnitude would amount to 4 space units. On the other hand, displacing C by 3 units would generate a feasible path for the robot as shown in Fig. 2c, and it would clearly be the optimal solution. This shows with a counterexample that the optimal solution to MOD cannot be obtained simply by juxtaposing optimal solutions to its sub-problems.

It is important to note that, in this work, we are not concerned about how objects can be displaced, both in terms of task or motion planning [5]. Neither are we concerned about the weight of an obstacle, or whether it would be necessary to push or pull it, in case of displacement. What we are concerned with is the amount by which obstacles must be displaced to yield a feasible path and assume that the computed displacements can be achieved by appropriate robot-obstacle physical interactions. From an algorithmic perspective, this assumption allows us to (1) assume a *planar*, 2D projection of the 3D environment, and (2) compute only the locations on such a plane where the obstacles are to be displaced. The same assumption allows us to consider events (both robot motions as well as object displacements) as *collapsed* to discrete time instants. If the robot took T discrete time instants to reach the goal location, with $0 \leq k \leq T$, we could as-

sociate both control actions and obstacle displacements (and zero displacements if an obstacle is not affected) to each time instant k in the sequence. In this work, we also disregard obstacle interactions, that is, the displaced obstacles are allowed to overlap with other obstacles, and therefore obstacle-obstacle collision avoidance is not considered.

The main contributions of the paper are:

- We prove that MOD is NP-hard even if all the obstacles are line segments by a reduction from the *Partition Problem*, which is NP-hard [6]. We further show that MOD remains NP-hard when the obstacles are restricted to axis-aligned rectangles. In addition, we establish the NP-hardness of MOD for circular obstacles. Together, these results substantially strengthen the NP-hardness result for the discrete MCD problem reported in [3].
- We present an exact approach to MOD by formulating the problem as a *Mixed Integer Quadratic Programming* problem. This approach returns exact solutions able to weight between path length and displacement amount but has worst-case exponential time.
- We introduce HS4MOD, an approximation method for MOD that divides MOD into m sub-problems and obtains a solution from a combination of optimal – yet local – solutions to these sub-problems. HS4MOD is shown to exhibit significant computational improvement over the exact approach even when the number of sub-problems is limited to two or three. For all the results presented in Section 6, we also compute the *optimality gap*, that is, the relative distance between the approximate and the optimal solution, with the aim of evaluating the quality of the solutions provided by HS4MOD.

The paper is organized as follows. We discuss related literature in Section 2.1. In Section 2.2, we provide a formal definition of the problem. We show in Section 3 that MOD is NP-hard for polygonal obstacles in a plane. Section 4 presents an exact mixed integer quadratic program optimization for MOD, and in Section 5 we introduce an approximate strategy for its solution. HS4MOD obtains a solution to MOD that combines optimal yet local solutions to its sub-problems. In Section 6, we evaluate HS4MOD under varying domains, distinct robot models, and different numbers of obstacles. The discussion in Section 7 concludes the paper.

2. Background and Problem Definition

2.1. Related Work

The problem we address in this paper shares some resemblance with a number of apparently similar problems discussed in the literature. Herein, we describe the related classes of problems highlighting similarities and differences, and we argue that MOD deserves a principled treatment on its own.

The MCD planner presented and discussed in [3] proceeds by (1) building a road map of robot configurations, and (2) sampling random obstacle displacements. The quality of the solution is improved by iteratively expanding the road map and the displacement samples. The method is shown to asymptotically approach the true optimum as the number of samples (both robot configurations and displacements) grow in size. In contrast, we formulate an exact approach, incorporating the robot kinematics and solve it find the optimal solution. Furthermore, we present an approximate method and compute its *deviation* from the computed optimal solution. MCD focuses on pure motion planning in the sense that it ignores robot dynamics and other differential constraints, and finds the translations and rotations required to move the robot. In our case, we plan a trajectory taking into account robot dynamics and other constraints such as control limits and collision constraints, to find the sequence of control actions to move the robot to the goal location.

The work in [7] examines minimum displacement planning for movable obstacles through a two-stage process. The first stage proceeds by finding a path through a set of movable obstacles while minimizing robot-obstacle overlaps. In the second stage, the overlapping obstacles are displaced iteratively by a distance factor until no overlap is found. However, the synthesized solutions are not necessarily optimal, and no guarantees or discussion regarding their quality are provided. Our approach computes the exact optimum and, for the proposed approximation algorithm, quantifies its solution quality with respect to the exact method across all experimental evaluations.

In a general sense, the problem we address is closely related to the *Navi-*

gation Among Movable Obstacles (NAMO) class of problems [8, 9, 10, 11, 12]. Most approaches for NAMO focus on connecting pairwise separated configuration regions by displacing objects as the robot proceeds to the goal location, and seek to find a motion plan that displaces the least number of obstacles while minimizing the *work* done to do it. Learning-based and language model-based approaches have also been explored for NAMO [13, 14]. However, these methods do not explicitly account for minimizing the displacement of manipulated objects during task execution.

NAMO problems are shown to be NP-hard if the final obstacle locations are unspecified, and PSPACE-hard when the locations are specified instead [15]. Stilman and Kuffner [8] address linear problems of degree 1 (LP_1), that is, two disjoint regions in the configuration space are linked by displacing a *single* obstacle. Subsequent studies [9, 10, 11] extend this concept to LP_k , where *at most* k obstacles are displaced to connect two free configuration regions. As such, NAMO relies on *keyhole* solutions, where a keyhole is the sub-problem of moving one (LP_1) or more objects (LP_k) to connect two regions of the free configuration space. Unlike these approaches, our method does not partition the configuration space into disconnected regions, nor does it impose restrictions on the number of obstacles that may be displaced. Though NAMO incorporates manipulation actions by selecting from obstacle contact points, the obstacles may be displaced unnecessarily far. Exact MOD tackles both displacement and path optimality concurrently.

Disconnection Proving in Motion Planning [16, 17, 18, 19] poses related challenges since it tries to find constraints preventing a robot to move from start to goal locations. Our approach does not need to consider explicitly hindering constraints, but adopts a horizon-slicing approximation, which requires connecting all the sub-optimal solutions.

The work in [20] considers a unified approach for solving constraint displacement problems (MOD, MCR, NAMO) by varying the objective function of a common optimization framework. However, the solutions obtained are suboptimal and lack guarantees with respect to optimality.

Manipulation Among Clutter or Rearrangement Planning [21, 22, 23, 24] is another related class of problems wherein objects may need to be displaced or rearranged to perform a given task. However, in these problems, the final location of the objects is either specified, or it could occur that displaced objects may be placed anywhere in the (free) workspace. Similarly, approaches for integrated *Task and Motion Planning* (TaMP) [25, 26, 27, 28, 29, 30] also bear resemblance to the problem discussed in this paper since task execution may require dealing with, at the motion planning level, constraints not considered at the task planning level. It is worth noting that the problem we consider here could be one of the key elements in such classes of problems.

The *Minimum Constraint Removal* problem (MCR) [31] finds the minimum number of movable obstacles that should be removed to yield a feasible path, but does not determine where the obstacles should be moved for creating such a path. Different algorithms for MCR exist in the literature [32, 33, 34, 35, 36], and the problem is proven to be NP-hard for convex polygonal obstacles [31]. Our formulation supersedes this class of approaches in that it determines also each object’s displacement.

2.2. The Minimum Obstacle Displacement Problem

Let us consider a generic robot, and let $\mathbf{x}_k$ denote the robot state at any time instant k . Let us consider, for the sake of the discussion, a scenario where a mobile robot should reach a goal location, and therefore $\mathbf{x}_k$ may denote its pose in terms of position and orientation. We assume a standard motion model for the robot given by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad (1)$$

where $\mathbf{u}_k$ is the control action applied at time instant k , where $k \in \{0, \dots, T-1\}$. Furthermore, let $\mathbf{x}^s$ and $\mathbf{x}^g$ denote the start and goal configurations, respectively. We shall denote the robot position at time k by $\mathbf{p}_k \subseteq \mathbf{x}_k$, and the sequence of all the positions from 0 to T as $\mathbf{p}$. Let

$$\mathcal{O} = \{o^i \mid 1 \leq i \leq |\mathcal{O}|\}$$

represent the set of obstacles in the environment, where $\mathcal{M} \subseteq \mathcal{O}$ is the set of movable obstacles. We will use $\mathbf{o}_k^i$ to denote the obstacle state associated with the obstacle o^i at time k . As it has been argued in the previous Section, at each time instant k , obstacles in $\mathcal{M}$ are associated with a displacement set

$$\mathcal{D}_k = \{d_k^i \in \mathbb{R}^n \mid 1 \leq i \leq \mathcal{M}\}$$

representing the corresponding obstacle displacements, where $d_k^i = 0$ in case the object $o^i \in \mathcal{M}$ is not displaced at time k . Both $\mathbf{x}_k$ and all obstacle locations $\mathbf{o}_k^i$ are, in principle, elements of the configuration space. It may be useful to define a function O that, when applied to the robot pose at time instant k , maps to the set of points corresponding to the robot body, namely $O(\mathbf{x}_k)$, and when applied to an obstacle pose $\mathbf{o}_k^i$, maps to the set of points corresponding to the obstacle itself, namely $O(\mathbf{o}_k^i)$. Before formally defining the problem addressed in this paper, we will rephrase the conceptual perimeter of this work:

- We do not focus on how objects are displaced;
- We do not take into account the weight of the obstacles, and instead we focus solely on their 2D projections;
- This work does not take into consideration collision avoidance among obstacles.
- We assume that a map of the environment is available a priori. Consequently, the proposed method is an offline planning approach that computes the required displacement for each obstacle before execution.

Now, we proceed to formally define the problem that is the focus of this paper.

Definition 1. The *Minimum Obstacle Displacement* (MOD) planning problem finds a sequence of control actions $\mathbf{u}_0, \dots, \mathbf{u}_{T-1}$ which make the robot navigate from $\mathbf{x}^s$ to $\mathbf{x}^g$ with obstacle displacement magnitudes $D^1, \dots, D^{\mathcal{M}}$ and is defined as

$$\min \quad w^x c(\mathbf{p}) + w^d \sum_{i=1}^{\mathcal{M}} c^i(D^i), \quad (2a)$$

$$\text{s.t.} \quad c(\mathbf{p}) = \sum_{k=0}^{T-1} \|\mathbf{p}_{k+1} - \mathbf{p}_k\|^2 + w^g \|\mathbf{p}_T - \mathbf{p}^g\|^2, \quad (2b)$$

$$O(\mathbf{x}_k) \cap \bigcup_{i=1}^{|\mathcal{O}|} O(\mathbf{o}_k^i) = \{\emptyset\}, \quad \forall k \in [1, T], \quad (2c)$$

$$c(D^i) = \sum_{k=0}^T d_k^i, \quad (2d)$$

$$\mathbf{x}_1 = \mathbf{x}^s. \quad (2e)$$

The objective function (2a) aims to minimize two components: the first component $c(\mathbf{p})$ is composed of two terms as given in (2b), which represent the length of the robot path while keeping track of the goal ($\mathbf{p}^g$) as terminal position, weighted by w^g ; in the second component, $c^i(D^i)$ are functions of displacement magnitudes; w^x , w^d are the respective weights for the two components. By tuning the weights in an appropriate way, it is possible to trade off path length and the magnitude of displacements, thereby allowing for shortest paths or more disruptive obstacle displacements. Condition (2c) guarantees that the robot does not intersect obstacles at any time. We note here that for obstacles with fixed position, $O(\mathbf{o}_k^i)$ remains constant throughout. Condition (2d) defines the total displacement of obstacles, and the initial condition is established in (2e).

It can be easily verified that when all displacements D^i are equal to 0, MOD solves the classical motion planning problem (see Section 6 for an example). Furthermore, MOD solves the shortest path problem if $c(\mathbf{p})$ measures the path length, and $w^d = w^x L^s$, where L^s is equal to the length of the shortest feasible path. MOD can also be employed to determine the minimum number of obstacles to be removed to find a feasible path, which is basically the MCR problem with constraints being obstacles. This is achieved by assigning $w^x = 0$ and limiting $d^i = \{0, 1\}$, that is, $d^i = 0$ for the obstacles being present and

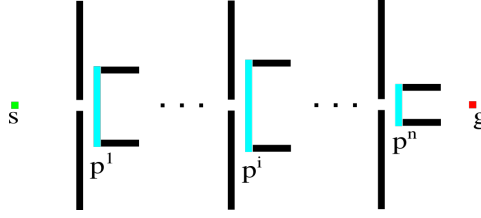


Figure 3: Construction by reduction from the *Partition* problem: movable line segments (in cyan) may be displaced only in the vertical direction. The segments in black are static obstacles. To pass along the x -axis (which connects start and goal locations) at location i , the obstacle at i has to be displaced at least by $p^i/2$. Otherwise, the robot has to move vertically for a distance of $p^i/2$ to reach one of the ends of obstacle at i , and then vertically by the same amount to reach again the x -axis.

$d^i = 1$ for the obstacles that are removed (an example is given in Section 6). It is noteworthy that by modifying $c^i(D^i)$ to incorporate the work done by the robot, MOD solves a variant of the NAMO problem [8]. Similarly, by relaxing w^d , MOD may be employed to solve variants of the re-arrangement planning problem.

Therefore, it can be deduced that MOD can generalize different classes of motion planning problems, and it deserves a treatment on its own.

3. Complexity of the MOD Problem

In this Section, we show that MOD is NP-hard for polygonal obstacles in the plane by considering two special cases, namely obstacles (1) as line segments, and (2) as axis-aligned rectangles with a unique displacement value. Furthermore, we also prove that MOD is NP-hard for circular-shaped obstacles.

3.1. MOD for Line Segments is NP-hard

We consider the class of problems in which obstacles are disjoint (vertical) line segments and do not touch each other. Therefore, there exists a feasible path (in the classical motion planning sense). However, it might be useful to displace some obstacles to obtain a shorter path. We consider the *decision* variant of the

problem, that is, if there is a feasible path to a goal location given a maximum path length of L^m and a maximum displacement *budget* D^m . This problem is NP-hard by reduction from the *Partition* problem, which is NP-complete [6].

We now prove the hardness. Given a set

$$P = \left\{ \frac{p^1}{2}, \dots, \frac{p^n}{2} \right\}$$

of n positive integers, the Partition problem checks whether two sets $P1$ and $P2 = P - P1$ exist such that $S(P1) = S(P2) = S(P)/2$, where $S(\cdot)$ computes the sum of the elements in the set. With reference to Fig. 3, we place the goal location $\mathbf{x}^g$ at a distance of $n + 1$ units along the x -axis from the start location $\mathbf{x}^s$. We have n line segments, whereas each line segment i is of length p^i , and is placed at a distance of i units from the start location. We set

$$D^m = \frac{S(P)}{2}$$

and

$$L^m = 2 \left(\frac{S(P)}{2} \right) + (n + 1).$$

Furthermore, we construct two vertical and two horizontal line segments as static obstacles close to each of the n movable obstacles. The fixed vertical line segments have a length of $2L^m + 1$ and are separated from each other by a distance $\delta = 1/4n$, which is assumed to be small enough for the robot to pass through. They are constructed to lie to the left of each movable obstacle at a distance of δ . The horizontal obstacle segments have a length of $(1 - 2\delta)$, and are placed touching the right end of each obstacle at i and lie symmetrically at a distance of $p^i/2$ from the x -axis. As a consequence of this construction, the set P can be partitioned into two sets $P1$ and $P2 = P - P1$ such that $S(P1) = S(P2)$ only if there is a path from $\mathbf{x}^s$ to $\mathbf{x}^g$ of length at most L^m , and displacement cost of at most D^m . Thus, MOD for (vertical) line segments in a plane is NP-hard.

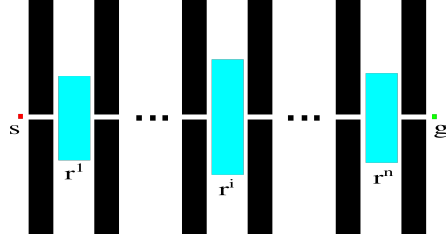


Figure 4: Movable rectangles are shown in cyan and can be displaced only by an amount r^i . Drawn in black are the obstacles constructed to achieve the reduction of the Partition problem.

3.2. MOD for Axis-aligned Rectangles with Constant Displacement Cost is NP-hard

In this configuration, each axis-aligned rectangle i is allowed a single displacement¹ of magnitude r^i . We consider again the decision version of the problem, that is, given a set of n movable obstacles in the form of axis-aligned rectangles, we want to ascertain if there is a feasible path from $\mathbf{x}^s$ to $\mathbf{x}^g$ such that the maximum path length and displacement do not exceed L^m and D^m , respectively. The NP-hardness can be proved by reduction from the Partition problem.

In order to prove the hardness, the scenario as shown in Fig. 4 is constructed. We place $\mathbf{x}^s$ and $\mathbf{x}^g$ at a distance of $2n + 1$ units along the x -axis. Each left end of a movable rectangle i is placed at $2i - 1$ units from $\mathbf{x}^s$. The rectangles have a length of r^i and a width of 1 unit, and are allowed to be displaced of r^i units in the vertical direction. At a distance of $2(i - 1) + \delta$ units from $\mathbf{x}^s$ we place two rectangular obstacles, vertically separated by a distance $\delta = 1/4n$, which is small enough for the robot to pass through. Each of these rectangles has a length of $2L^m + 1$ and a width of $1 - 2\delta$. Now, let us consider a set

¹We note here that while the result presented is of theoretical interest, in the real world the single-displacement value considered here can be mapped to obstacles moving to a specified final location – for example, the transportation of pallets in logistics [37].

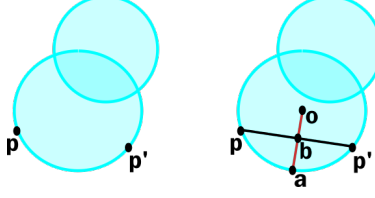


Figure 5: Construction for NP-hardness proof of MOD with circular obstacles. See text for further details.

$R = \{r^1, \dots, r^n\}$ and let

$$D^m = \frac{S(R)}{2}$$

and

$$L^m = \frac{S(R)}{2} + (2n + 1).$$

The set R can be partitioned into two sets $R1$ and $R2 = R - R1$ such that $S(R1) = S(R2)$ only if there is a path from $\mathbf{x}^s$ to $\mathbf{x}^g$ of length at most L^m , and displacement cost of at most D^m . Thus, the problem of deciding if there is a path with length at most L^m and displacement cost at most D^m is NP-hard.

3.3. MOD for Circular Obstacles is NP-hard

Let us consider a scenario where circular obstacles are located on the plane. When the robot encounters an obstacle at point p (refer to Fig. 5), it navigates along the arc $\widehat{pp'}$ if the obstacle remains stationary, with p' denoting the point where the robot departs from the obstacle's boundary. If the obstacle is displaced, the shortest path $\overline{pp'}$ for the robot is achieved by displacing the obstacle at least by an amount $d = \overline{ab}$, with $\overline{oa} \perp \overline{pp'}$, and o being the center of the obstacle. Thus, for a given displacement d , the shortest path $\overline{pp'}$ may easily be computed as

$$\overline{pp'} = 2\sqrt{r^2 - (r - d)^2},$$

with r denoting the radius of the obstacle. If the obstacle is displaced by an amount d , the robot path length decreases by an amount $\widehat{pp'} - \overline{pp'}$. For the discussion that follows, the decrease in the path length for a given d will be referred to as the *value* associated with d .

We now establish the NP-hardness of the MOD problem by a polynomial-time reduction from the *Multiple-Choice Knapsack Problem* (MCKP), which is NP-hard [38]. We consider the budget-constrained discrete special case of MOD, in which a finite set of admissible displacements is specified for each movable obstacle and the total obstacle displacement is bounded by a given budget D^m .

In an instance of MCKP, there are n mutually disjoint classes $M^1, \dots, M^n$ of items and a knapsack of capacity c . Each item $j \in M^i$ has a weight W^{ij} and a profit P^{ij} . The objective is to select exactly one item from each class such that $\sum_{i=1}^n W^{ij_i} \leq c$, while maximizing $\sum_{i=1}^n P^{ij_i}$.

We construct an instance of the budget-constrained discrete MOD problem with n movable circular obstacles, where obstacle i corresponds to class M^i . For every item $j \in M^i$, we construct one admissible displacement d^{ij} and its corresponding path-length reduction v^{ij} . The construction is chosen such that $d^{ij} = W^{ij}$, $v^{ij} = P^{ij}$. The latter correspondence is realizable from the circular-obstacle geometry since the path-length reduction produced by a displacement d of a circular obstacle of radius r_i is $V_i(d) = 2\sqrt{r_i^2 - (r_i - d)^2} = 2\sqrt{2r_id - d^2}$.

For a prescribed value P^{ij} , an appropriate radius and displacement can be selected so that $V_i(d^{ij}) = P^{ij}$. In particular, the geometric parameters can be constructed using the inverse relation $r_i = \frac{(P^{ij})^2 + 4(d^{ij})^2}{8d^{ij}}$, with the construction chosen consistently for the admissible displacement set of obstacle i . We then set the maximum total displacement to $D^m = c$. Consequently, selecting one displacement for each obstacle gives $\sum_{i=1}^n d^{ij_i} \leq D^m \iff \sum_{i=1}^n W^{ij_i} \leq c$, while maximizing the total path-length reduction gives $\max \sum_{i=1}^n v^{ij_i} \iff \max \sum_{i=1}^n P^{ij_i}$.

Thus, every feasible selection of one item from each MCKP class corresponds to a feasible selection of one displacement for each obstacle, with identical objective value, and vice versa. Hence, an optimal solution of the constructed MOD instance yields an optimal solution of the original MCKP instance. The construction requires only a polynomial number of obstacle parameters and displacement choices, and therefore can be performed in polynomial time. Hence, the budget-constrained discrete special case of MOD, and consequently MOD

itself, is NP-hard.

4. An Exact Approach to Solve MOD

In this Section, we present an exact approach to solve MOD that achieves an optimal, global solution, albeit computationally expensive as the set of obstacles $\mathcal{O}$ and the robot workspace grow.

4.1. The Optimization Problem

Using the models related to robots and obstacles, we formulate MOD as an optimization problem that finds a feasible path for the robot to the goal location while minimizing a trade off between path length and obstacle displacements. While doing so, control inputs and state variables must lie within their respective feasible sets, and the robot should not collide with obstacles. Starting from the general formulation of the MOD problem, the overall optimization problem can be thus formalized as follows.

$$\min \quad J = w^x c(\mathbf{p}) + w^d \sum_{i=1}^{\mathcal{M}} c^i(D^i), \quad (3a)$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad (3b)$$

$$\mathbf{o}_{k+1} = g(\mathbf{o}_k, \mathbf{d}_k), \quad (3c)$$

$$\underline{\mathbf{x}} \leq \mathbf{x}_k \leq \bar{\mathbf{x}}, \quad (3d)$$

$$\underline{\mathbf{u}} \leq \mathbf{u}_k \leq \bar{\mathbf{u}}, \quad (3e)$$

$$\underline{\mathbf{o}} \leq \mathbf{o}_k \leq \bar{\mathbf{o}}, \quad (3f)$$

$$\underline{\mathbf{d}} \leq \mathbf{d}_k \leq \bar{\mathbf{d}}, \quad (3g)$$

$$O(\mathbf{x}_k) \cap \bigcup_{i=1}^{|\mathcal{O}|} O(\mathbf{o}_k^i) = \{\emptyset\}, \quad \forall k \in [0, T], \quad (3h)$$

$$\forall k \in \{1, \dots, T\},$$

where an obstacle at time k , denoted by o_k , is displaced by $\mathbf{d}_k^i$ according to some function $g(\cdot)$, which denotes the employed robot strategy to move away objects. The robot and obstacle dynamics are represented in constraints (3b)-(3c) subject to the upper and lower bounds on the state and control variables as given in (3d)-(3g). Finally, constraint (3h) represents the collision avoidance constraint between the robot and the obstacles.

4.2. Obstacles' Model

As we noted before, we are not concerned about *how* obstacles are displaced by a robot, and we assume that they can be re-positioned irrespective of the nature of the required robot-obstacle interaction. In the formulation of the optimization problem, this is encoded in a generic function $g(\cdot)$. Therefore, an obstacle may be moved without any constraints. Each obstacle is capable of translation and rotation and hence the state $\mathbf{o}^i = [o^{i,x}, o^{i,y}, o^{i,\theta}]$ includes translations in the x and y directions, as well as a rotation. This yields the following *linear* model for an obstacle o^i :

$$\begin{aligned} o_{k+1}^{i,x} &= o_k^{i,x} + d_k^{i,x}, \\ o_{k+1}^{i,y} &= o_k^{i,y} + d_k^{i,y}, \\ o_{k+1}^{i,\theta} &= o_k^{i,\theta} + d_k^{i,\theta}, \end{aligned} \tag{4}$$

where $\mathbf{d}_k^i = [d_k^{i,x}, d_k^{i,y}, d_k^{i,\theta}]$ specify the obstacle's displacement between time instants k and $k+1$. Thus, the displacement between two time instants for the obstacle o^i is $d_k^i = \|\mathbf{d}_k^i\|$. In analogy to the standard motion model introduced in (1), we may write (4) compactly as $\mathbf{o}_{k+1} = g(\mathbf{o}_k, \mathbf{d}_k)$.

We note that, in many warehouse, logistics, and other structured environments, the placement regions for movable objects are predefined. In such scenarios, these placement regions can be directly treated as displacement sets from which the object configurations are sampled. This eliminates the need to explicitly formulate the obstacle model as described above, thereby reducing the solution search space and, consequently, the computational complexity of the problem.

4.3. Robot's Model

We do not assume any specific robot motion model for the validity of our work, and we refer to any standard motion model as in the form given in (1). However, as it will become clear in later Sections, we distinguish between linear and nonlinear robot models. In this paper, we consider two different linear models. The first is a trivial model similar to the obstacle model in (4), such as

$$\begin{aligned}x_{k+1} &= x_k + \tau u_k^x, \\y_{k+1} &= y_k + \tau u_k^y, \\\theta_{k+1} &= \theta_k + \tau u_k^\theta,\end{aligned}\tag{5}$$

with $\mathbf{x}_k = [x_k, y_k, \theta_k]$ as the robot pose and $\mathbf{u}_k = [u_k^x, u_k^y, u_k^\theta]$ as the applied control at time instant k . We also consider the linear dynamics model borrowed from [39] with a robot state $\mathbf{x}_k = [x_k, y_k, v_k^x, v_k^y]$ consisting of its location and velocity, and with acceleration input $\mathbf{u}_k = [a_k^x, a_k^y]$. The corresponding model is

$$\begin{aligned}x_{k+1} &= x_k + \tau v_k^x + \frac{\tau^2}{2} a_k^x, \\y_{k+1} &= y_k + \tau v_k^y + \frac{\tau^2}{2} a_k^y, \\v_{k+1}^x &= v_k^x + \tau a_k^x, \\v_{k+1}^y &= v_k^y + \tau a_k^y,\end{aligned}\tag{6}$$

where τ is, again, the duration of a time step. The use of linear models allows for obtaining optimal, global solutions, albeit dependent on the nature of the objective function as well as other constraints. However, the motion model of mobile robots is often nonlinear. Nonlinear and non-convex models provide weak theoretical guarantees and the optimization often results in local minima. Nevertheless, we evaluate our approach on two different nonlinear models. The first model is a simple differential drive model [1] given by

$$\begin{aligned}x_{k+1} &= x_k + \tau u_k \cos \theta_k, \\y_{k+1} &= y_k + \tau u_k \sin \theta_k, \\\theta_{k+1} &= \theta_k + \tau \omega_k,\end{aligned}\tag{7}$$

where $\mathbf{x}_k = [x_k, y_k, \theta_k]$ is the robot state and $\mathbf{u}_k = [u_k, \omega_k]$ is the applied control at time instant k . We then consider the case of a non-holonomic car-like robot [40]

$$\begin{aligned} x_{k+1} &= x_k + \tau v_k \cos \theta_k, \\ y_{k+1} &= y_k + \tau v_k \sin \theta_k, \\ \theta_{k+1} &= \theta_k + \tau v_k \frac{\tan \phi_k}{l}, \\ v_{k+1} &= v_k + \tau a_k, \end{aligned} \tag{8}$$

where $\mathbf{x}_k = [x_k, y_k, \theta_k, v_k]$ is the robot state consisting of its pose and speed, $\mathbf{u}_k = (a_k, \phi_k)$ is the applied control consisting of the acceleration and steering wheel angle, τ is the duration of a time-step, and l is the length of the robot.

4.4. Collision Avoidance

So far we have been agnostic to the structure of the constraint in (3h). In the following paragraphs, we formulate the collision avoidance constraint more precisely. We will consider two such constraints, namely (1) polygon-shaped obstacles, and (2) circle-shaped obstacles.

4.4.1. Polygonal Obstacles

A polygonal obstacle can be represented as a polyhedron $\mathcal{P}$, that is, the intersection of $N \in \mathbb{N}$ halfspaces, such that

$$\mathcal{P} = \{x \in \mathbb{R}^2 | Ax \leq b\}, \tag{9}$$

where $A = (a^1, \dots, a^N)^T$ and $b = (b^1, \dots, b^N)^T$. Each row $a^i x \leq b^i$ defines one of the halfspaces of $\mathcal{P}$.

For this specific shape, the collision avoidance constraint can be written by means of a mixed integer formulation as

$$a^i (\mathbf{p}_k - \mathbf{o}_k^i) \geq b^i - c_k^i M, \quad i = 1, \dots, N \tag{10}$$

$$\sum_{i=1}^N c_k^i \leq N - 1, \tag{11}$$

where $c_k^i \in \{0, 1\}$ are binary auxiliary variables and M is a constant suitably large. It is important to note that (10) and (11) holds $\forall k \in \{1, \dots, T-1\}$ and $\forall i \in \{1, \dots, |\mathcal{O}|\}$.

4.4.2. Circular Obstacles

It is common practice to model robot and obstacles using approximate bounding volume spheres [41, 42]. In such cases, the collision avoidance constraint is given by

$$\|\mathbf{p}_k - \mathbf{o}_k^i\| \geq r^r + r^o, \quad (12)$$

where $\mathbf{p}_k, \mathbf{o}_k^i$ are the midpoints of the robot and obstacle circles (that is, the 2D projections of the sphere on the supporting plane), respectively, whereas r^r and r^o are the corresponding radii.

When the robot and all the obstacles are modeled as circles, the collision avoidance constraint in (12) can be linearized by introducing j binary auxiliary variables $\xi_k^{j,i} \in \{0, 1\}$ for each obstacle $\mathbf{o}^i \in \mathcal{O}$ at every time instant k . The constraint (12) is thus rewritten as

$$x_k - o_k^{i,x} \geq r^r + r^o - \xi_k^{1,i} M, \quad (13)$$

$$-(x_k - o_k^{i,x}) \geq r^r + r^o - \xi_k^{2,i} M, \quad (14)$$

$$y_k - o_k^{i,y} \geq r^r + r^o - \xi_k^{3,i} M, \quad (15)$$

$$-(y_k - o_k^{i,y}) \geq r^r + r^o - \xi_k^{4,i} M, \quad (16)$$

$$\sum_{j=1}^4 \xi_k^{j,i} \leq 3, \quad (17)$$

where it holds $\forall k \in \{1, \dots, T-1\}$ and $\forall i \in \{1, \dots, |\mathcal{O}|\}$, and M is a suitably large constant.

As a result of (10), (11), (13)-(17) the optimization problem in (3) can be effectively converted into a *Mixed Integer Quadratic Programming* (MIQP) problem. MIQPs are a class of optimization problems with a quadratic objective function subject to linear constraints, and involving both continuous and discrete decision variables. It is well known from Operations Research theory that

there are algorithms guaranteeing global optimality for this type of optimization problem.

In case of nonlinear robot dynamics, the optimization problem (3) is an instantiation of *Nonlinear Programming* (NLP). However, due to the non-convex nature of the NLP formulation, it is not possible to guarantee a global optimum [43]. If we choose to reformulate the collision avoidance constraints by introducing auxiliary variables (as shown in (10)-(11) and (13)-(17), the optimization problem transforms into a *Mixed Integer Nonlinear Programming* (MINLP) problem. This change in problem class increases complexity further due to the combinatorial aspects introduced by the use of binary variables. Moreover, global optimization algorithms for MINLPs often necessitate convexity in both the objective and constraint functions [44]. Therefore, for nonlinear robot models, we resort to solving the NLP formulation to obtain sub-optimal solutions.

5. A Horizon-Slicing Heuristic to Solve MOD

In the previous Section, we have modeled MOD as an optimization problem, as formulated in (3). The exact solution method does not solve all problem instances efficiently, because of its inherent intractability, and it seems beneficial to search for approximate solutions. This is the case in many practical situations in which we are concerned with obtaining a feasible yet sub-optimal solution that has the major advantage of being computed easily. As discussed above, in MIQP problems, the worst-case computational complexity grows exponentially with the number of binary and/or integer decision variables [45]. As a case in point, consider the linearisation of (12), where for each obstacle we have introduced four binary variables (13)-(16) to formulate the optimization as an MIQP problem when the robot dynamics is linear. The number of feasible combinations of the binary variables is thus $2^{4 \times T \times |\mathcal{O}|}$, where T is the number of time steps required to reach the goal location. With the increase in the number of movable obstacles and the environment *size* (in the sense that the

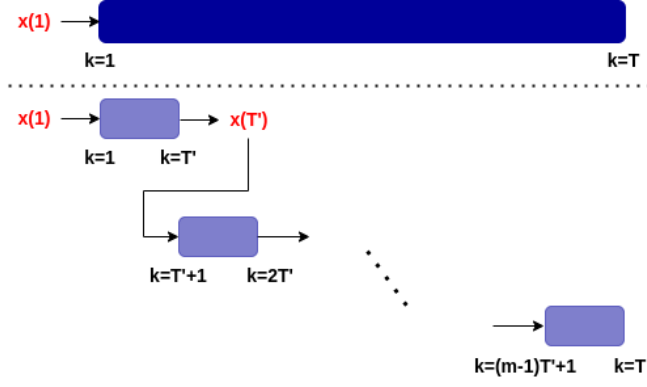


Figure 6: An illustration of the horizon-slicing approach employed in this work. $x(\cdot)$ denotes the variables of interest. The input (initial state) to the original problem and the first slice of the sub-problem coincide, and are referred to as $x(1)$. The output of the first slice $x(T')$ is the initial state for the next slice. The values of the time-dependent variables are passed on to the slices that follow.

larger the environment, the higher the value of T on average), the optimization process tends to be computationally unfeasible. The computational burden could be reduced by decreasing the number of variables. One *naïve* way to achieve this result is by progressively solving subsets of the overall problem, each subset being computationally less expensive compared to solving the complete optimization problem at once. However, we have already seen that MOD does not exhibit the optimal substructure property. Therefore, solving subsets of the whole problem does not necessarily guarantee an optimal solution at the global level.

5.1. HS4MOD

In order to reduce the computational effort (at the expense of giving up optimality) we temporally partition the whole optimization problem into m sub-problems, each one of duration T' such that $mT' = T$. In this paper, we simply *slice* the number of time steps required to reach the goal location into a smaller duration, and solve the optimization problem for each slice separately. Once each sub-problem undergoes a specific optimization process, the values of

time-dependent variables are passed on to the sub-problem that follows. This concept is illustrated in Fig. 6. It should be pointed out that different, *probably smarter* strategies to horizon-slicing may be employed, for instance related to the adopted motion planning approach. A comprehensive analysis of such strategies is outside the scope of this paper, whereas the adopted horizon-slicing approach serve the purpose of allowing us to draw a number of general conclusions about sub-optimality and time complexity, which can be used as prerequisite work to ground further strategies.

Let us consider the objective function J in (3a), and let us denote the optimal objective function *without* horizon-slicing as J^* . We thus have:

$$\begin{aligned}
J^* &= w^x c(\mathbf{p}) + w^d \sum_{i=1}^{\mathcal{M}} c^i(D^i) \\
&= \sum_{k=0}^{T-1} w^x \|\mathbf{p}_{k+1} - \mathbf{p}_k\|^2 + w^x w^g \|\mathbf{p}_T - \mathbf{p}^g\|^2 + \\
&\quad w^d \sum_{i=1}^{\mathcal{M}} \sum_{k=0}^T d_k^i \\
&= \sum_{k=0}^{T-1} \left(w^x \|\mathbf{p}_{k+1} - \mathbf{p}_k\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_k^i \right) + \\
&\quad w^x w^g \|\mathbf{p}_T - \mathbf{p}^g\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_T^i.
\end{aligned} \tag{18}$$

Let us now focus on the terms inside the bracket. This sum can be re-written

as

$$\begin{aligned}
& \sum_{k=0}^{T-1} w^x \|\mathbf{p}_{k+1} - \mathbf{p}_k\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_k^i \\
&= \left[\sum_{k=0}^{T'-1} w^x \|\mathbf{p}_{k+1} - \mathbf{p}_k\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_k^i \right] + \\
& \quad \dots + \\
& \quad \left[\sum_{k=(m-1)T'}^{T-1} w^x \|\mathbf{p}_{k+1} - \mathbf{p}_k\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_k^i \right] \\
&= J_{0 \rightarrow T'-1}^{\star, s} + \dots + J_{(m-1)T' \rightarrow T-1}^{\star, s} \\
&= \sum_{j=1}^m J_{(j-1)T' \rightarrow jT'-1}^{\star, s},
\end{aligned} \tag{19}$$

where

$$J_{(j-1)T' \rightarrow jT'-1}^{\star, s} = \sum_{k=(j-1)T'}^{jT'-1} \left(w^x \|\mathbf{p}_{k+1} - \mathbf{p}_k\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_k^i \right), \tag{20}$$

and m is the number of slices. Thus, (18) can be written as

$$J^{\star} = \sum_{j=1}^m J_{(j-1)T' \rightarrow jT'-1}^{\star, s} + w^x w^g \|\mathbf{p}_T - \mathbf{p}^g\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_T^i. \tag{21}$$

Let us now consider the scenario wherein we partition the horizon and address m distinct optimization problems. For each horizon slice j , the corresponding objective function is formulated as

$$\begin{aligned}
J^j &= \sum_{k=(j-1)T'}^{jT'-1} \left(w^x \|\mathbf{p}_{k+1} - \mathbf{p}_k\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_k^i \right) + \\
& \quad w^x w^g \|\mathbf{p}_{jT'-1} - \mathbf{p}^g\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_{jT'-1}^i.
\end{aligned} \tag{22}$$

Let us define $\eta^j = w^x w^g \|\mathbf{p}_{jT'-1} - \mathbf{p}^g\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_{jT'-1}^i$. Using the simplification employed in (19), (22) can be written as

$$J^j = J_{(j-1)T' \rightarrow jT'-1}^{j, s} + \eta^j. \tag{23}$$

Adding the objective function for the m horizon slices and using the fact that

$$\eta^m = w^x w^g \|\mathbf{p}_T - \mathbf{p}^g\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_T^i, \quad (24)$$

we have

$$\begin{aligned} J &= \sum_{j=1}^m \left(J_{(j-1)T' \rightarrow jT'-1}^{j,s} + \eta^j \right) \\ &= \sum_{j=1}^m J_{(j-1)T' \rightarrow jT'-1}^{j,s} + \sum_{j=1}^{m-1} \eta^j + \\ &\quad w^x w^g \|\mathbf{p}_T - \mathbf{p}^g\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_T^i. \end{aligned} \quad (25)$$

Algorithm 1 HS4MOD

Input: $T, T', m, \mathbf{x}^s, \mathbf{x}^g, \mathcal{O}, \mathcal{M}, w^x, w^d, w^g, f(\cdot), g(\cdot)$

```

1: for  $j = 1, \dots, m$  do
2:    $J^j \leftarrow 0$ 
3:   for  $k = (j-1)T', \dots, jT'-1$  do
4:      $J^j \leftarrow J^j + w^x \|\mathbf{p}_{k+1} - \mathbf{p}_k\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_k^i$ 
5:      $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$ 
6:      $\mathbf{o}_{k+1} = g(\mathbf{o}_k, \mathbf{d}_k)$ 
       such that (3b)-(3h)
7:   end for
8:    $J^j \leftarrow J^j + w^x w^g \|\mathbf{p}_{jT'-1} - \mathbf{p}^g\|^2 + w^d \sum_{i=1}^{\mathcal{M}} d_{jT'-1}^i$ 
9:   Minimize  $J^j$  such that (3b) – (3h)
10: end for
```

Output: $J, \{\mathbf{x}_0, \dots, \mathbf{x}_T\}, \{d_k^i\} \forall i \in \mathcal{M} \text{ and } \forall k \in \{1, \dots, T\}$

It can be readily verified that equation (25) exhibits a structure similar to (19), but with an added terminal cost for each slice except the final one. This additional term is present in each sub-problem as it enables the robot to keep track of the goal. The overall HS4MOD algorithm is summarized in Algorithm 1. For each slice j the exact optimization is performed (lines 2-7).

The summation in (22) is carried out (as shown in line 4), and the goal tracking term is added (line 8) to obtain the objective function J^j (22) for each slice. The optimization is then carried out subject to the constraints given in (3b)-(3h).

5.2. Analysis of the Approximate Solution

In order to establish a meaningful comparison between J and J^* , we introduce a new objective function, denoted as J' , which aggregates the cost obtained by summing up the costs of all the m individual slices, and is thus given by

$$J' = J - \sum_{j=1}^{m-1} \eta^j. \quad (26)$$

J' has a structure similar to that of J^* . As observed before, from an algorithmic perspective, a subset of an optimal solution is not necessarily globally optimal for the particular problem we consider. However, it can be easily verified that the solutions to all the subsets are naturally feasible solutions. Therefore, the horizon-slicing approach results in a feasible solution to the original problem, and therefore the value of J' must be an upper bound to J^* , that is, $J' \geq J^*$. For this reason, we have that

$$\begin{aligned} J' &= \sum_{j=1}^m J_{(j-1)T' \rightarrow jT'-1}^{j,s} + \eta^m \\ &= \sum_{j=1}^m \left(J_{(j-1)T' \rightarrow jT'-1}^{*,s} + \delta^i \right) + \eta^m \\ &\leq \sum_{j=1}^m \left(J_{(j-1)T' \rightarrow jT'-1}^{*,s} + \delta^{max} \right) + \eta^m \\ &\leq \left(\sum_{j=1}^m J_{(j-1)T' \rightarrow jT'-1}^{*,s} + \eta^m \right) + m\delta^{max} \\ &\leq J^* + m\delta^{max}, \end{aligned} \quad (27)$$

where

$$\delta^i = J_{(j-1)T' \rightarrow jT'-1}^{j,s} - J_{(j-1)T' \rightarrow jT'-1}^{*,s}, \quad (28)$$

with each δ^i bounded by $\pm\delta^{max}$, that is

$$\delta^i \in [-\delta^{max}, +\delta^{max}]. \quad (29)$$

From (27), we thus have

$$J^* \leq J' \leq J^* + m\delta^{max}. \quad (30)$$

Let the deviation from the optimal solution $J' - J^*$ be a factor of J^* , that is, $J' - J^* = \epsilon J^*$. Ideally, we would want $\epsilon \rightarrow 0$ with considerable computational efficiency. We note here that, in optimization literature, the standard metric used to compute the relative distance between an approximate and the optimal solution is the *optimality gap*. Essentially, ϵ computes this relative distance and is therefore equivalent to the optimality gap since

$$\text{optimality gap} = \frac{J' - J^*}{J^*} = \frac{\epsilon J^*}{J^*} = \epsilon. \quad (31)$$

From (30), we have

$$J' - J^* \leq m\delta^{max} \implies \epsilon J^* \leq m\delta^{max} \quad (32)$$

and therefore

$$\epsilon \leq m \frac{\delta^{max}}{J^*}. \quad (33)$$

This relation characterizes the relative deviation for a given problem instance; it should not be interpreted as an a priori approximation guarantee. Clearly, for a given problem, as the number of horizon slices or sub-problems increase, the value of ϵ grows as well. The exact deviation ϵ for a given m is problem-specific (see Fig. 7) since δ^{max} depends on a number of parameters such as the robot and obstacle dynamics, size of the environment, distribution of the obstacles in the environment, number of slices. In Section 6, we provide an empirical analysis to determine the best m by performing different experiments while varying some of the parameters stated above.

6. Evaluation and Discussion

In this Section, we evaluate our approach using (1) the optimal but computationally demanding exact approach discussed in Section 4, and (2) the HS4MOD sub-optimal approach, which was introduced in Section 5. The optimization

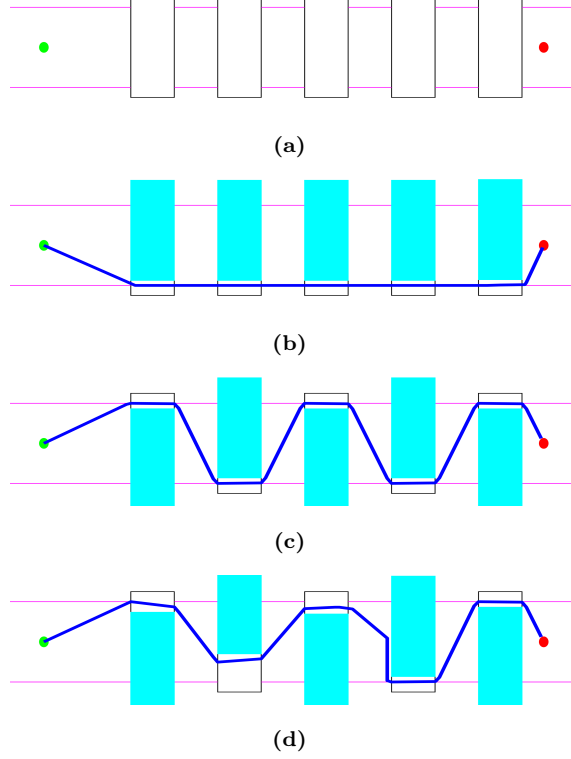


Figure 7: (a) A robot is tasked to move from the left (start location in green) to the right-hand side (goal location in red) in the presence of 5 movable obstacles. The y -coordinate of the robot is constrained to lie within the magenta lines. (b) Obstacles can only translate in the vertical direction. The cases $m = 1$ and $m = 2$ produce similar trajectories (obstacle displacements shown in cyan). (c)-(d) Odd obstacles (counting from left) can only move downward and even obstacles can only move upward. The optimal solution is given in (c). The case $m = 2$ gives a sub-optimal solution (with $\epsilon = 0.81$) as shown in (d) with the first sub-problem involving the first 3 obstacles, as shown by the vertical trajectory after the third obstacle.

problem in (3) is performed using the IBM ILOG CPLEX Optimization Studio V12.10.0 in MATLAB under the YALMIP interface [46]. When the robot model is nonlinear, optimization is achieved employing the IPOPT solver [47]. In both cases, the time limit has been set up to 1000 seconds. The performance

is evaluated on an Intel® i7-10850H CPU @ 2.70GHz with 32 GB RAM under Windows 11. The horizon T was set between 23 and 29 for the SQUARE and RANDOM environments and to 50 for the ROBOT environments. All experiments were conducted in MATLAB/YALMIP using CPLEX with its default relative and absolute MIP gaps (10^{-4} and 10^{-6}) and IPOPT with an overall tolerance of 10^{-7} , feasibility and complementarity tolerances of 10^{-4} , and acceptable termination after 15 consecutive iterations satisfying a tolerance of 10^{-6} . The objective weights were selected to prioritize goal attainment, followed by displacement minimization.

HS4MOD has been tested on three different domains, namely, (1) a SQUARE domain, of size 24×24 meters, with 36 obstacles — we consider two domains with the same obstacle configurations but different goal locations, naming them as SQUARE_1 (see Fig. 8a) and SQUARE_2 (Fig. 8b), (2) a ROBOT domain, of size 79×18 meters, with 74 obstacles (Fig. 8c), and (3) a RANDOM domain, of size 24×24 meters, with 100 obstacles (Fig. 8d). The robot must move from a start location (marked in green in Fig. 8) to a goal location (marked in red) in the presence of movable circular obstacles (in cyan). As discussed in Section 4, m denotes the number of horizon slices and $m = 1$ corresponds to the *exact* approach. At present, the values of w^g , w^x , and w^d are determined through empirical tuning. Unless otherwise specified, we use $w^g = 10$, $w^x = 0.5$, and $w^d = 100$ for all the experiments.

6.1. Linear Robot Model

We first test HS4MOD adopting the robot dynamics in (5). The robot trajectory and the computed obstacle displacements are visualized in Fig. 9 and in Fig. 10. Obstacles that have been displaced by the robot passage are characterized by a shaded trail of previous locations, whereas the cyan color represents their final location after the displacement. Various statistics for the same values are presented in Table 2 under the robot model (5) column. The exact optimization case is denoted by $m = 1$, and shows substantial computational load, as seen from Table 2 — the first line of each domain. As it has been argued

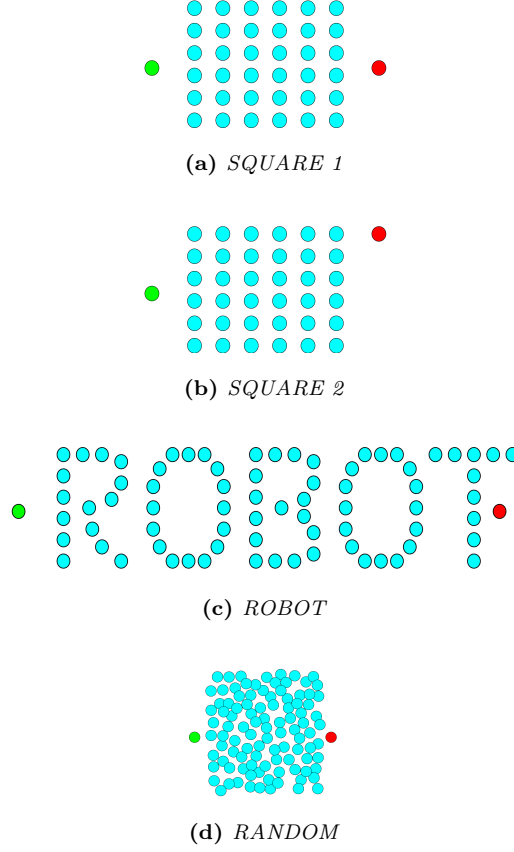


Figure 8: Different domains are used to test our methods. In all the experiments, a circular robot is asked to move from the left (start location in green) to the right (goal location in red) hand side in the presence of movable circular obstacles (in cyan).

before, the computational complexity is exponential in the number of binary variables, which is proportional to the number of movable obstacles, and it is particularly evident from the computational time in the *ROBOT* and *RANDOM* domains, with 74 and 100 obstacles, respectively. The complexity depends also on (1) the spread of the obstacles with respect to the start and goal locations, and (2) the size of the environment. The first factor accounts for the difference in computation time for the cases $m = 1$ and $m = 2$ in the *SQUARE_1* and *SQUARE_2* domains, despite having the same number of binary variables. The

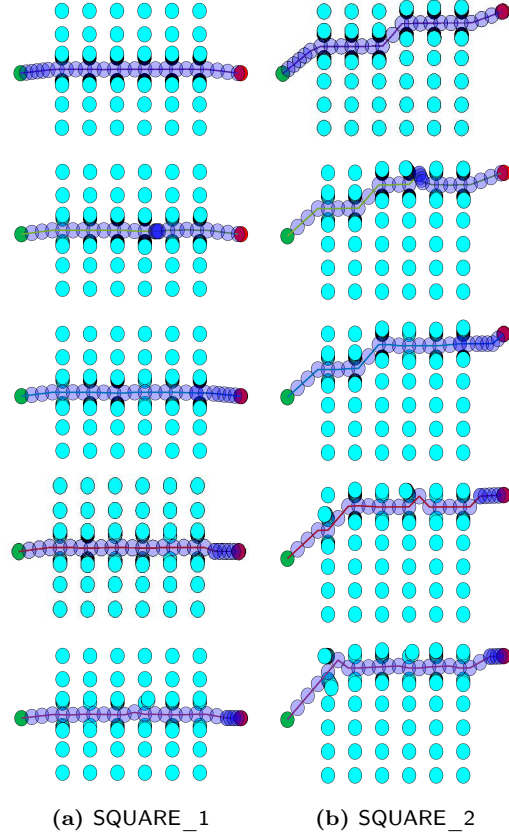


Figure 9: Robot trajectories and obstacle displacements for SQUARE_1 and SQUARE_2 while the robot moves according to the model in (5). The rows are related to different values for m , the top row being related to $m = 1$ (exact solution), the second row to $m = 2$, and so on.

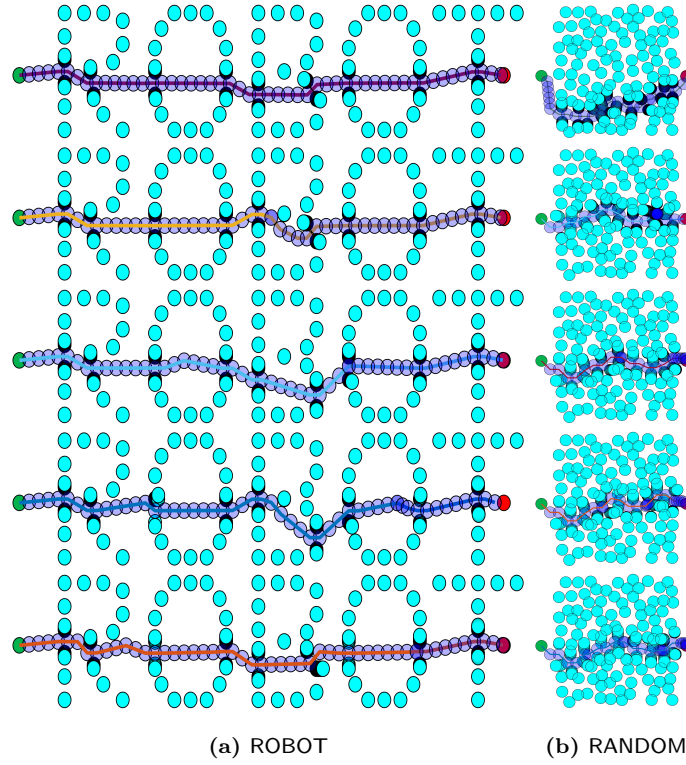


Figure 10: Robot trajectories and obstacle displacements for ROBOT and RANDOM while the robot moves according to the model in (5). The rows are related to different values for m , the top row being related to $m = 1$ (exact solution), the second row to $m = 2$, and so on.

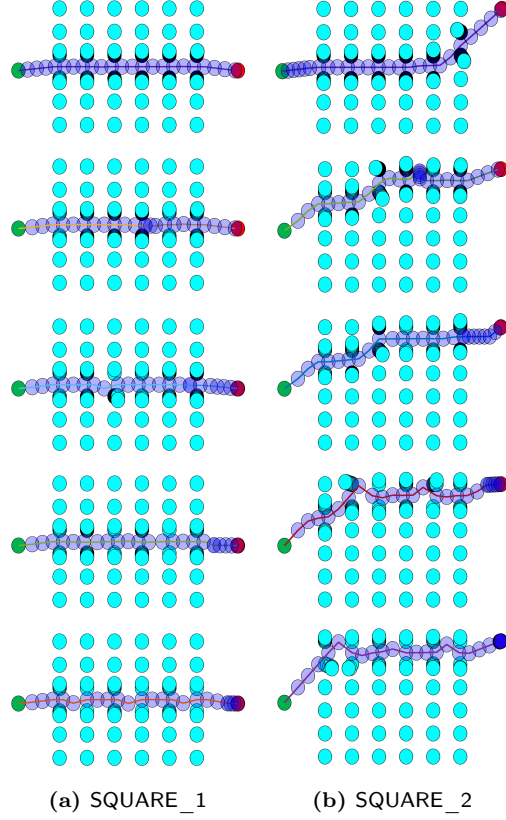


Figure 11: Robot trajectories and obstacle displacements for `SQUARE_1` and `SQUARE_2` while the robot moves according to the model in (6). The first row corresponds to $m = 1$ (no horizon slicing) for all four domains, the second row corresponds to $m = 2$, and so on.

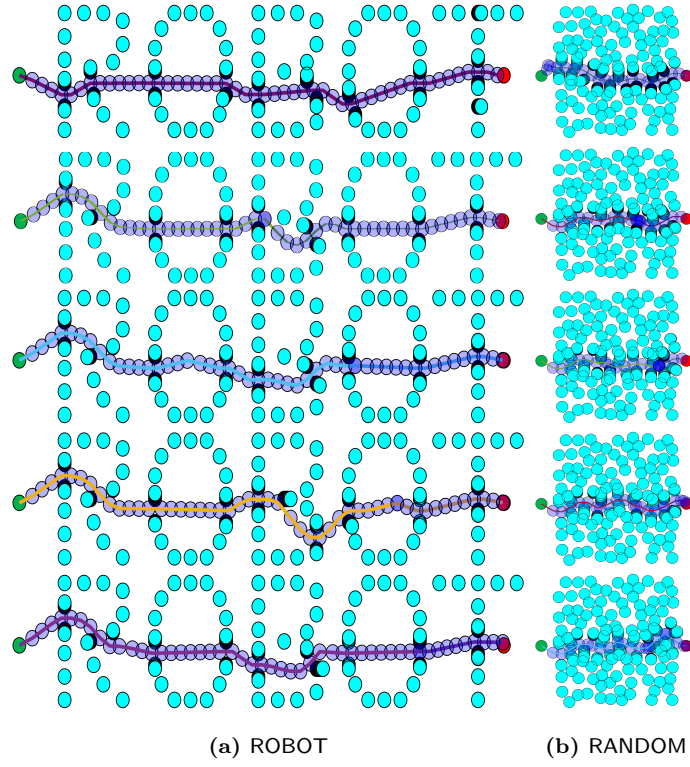


Figure 12: Robot trajectories and obstacle displacements for ROBOT and RANDOM while the robot moves according to the model in (6). The first row corresponds to $m = 1$ (no horizon slicing) for all four domains, the second row corresponds to $m = 2$, and so on.

two factors yield a higher computational time for the ROBOT domain than the RANDOM domain. Analogous results can be seen when employing the linear robot model in (6) — the robot trajectory and obstacle movements visualized in Fig. 11 and Fig. 12. Clearly, in all domains, for both robot models, as m increases, the deviation from the optimal solution increases as well thereby resulting in sub-optimal solutions as corroborated by the rise in ϵ values. At the same time, higher values of m result in great computational efficiency. Keeping in mind a pragmatic trade-off between optimality and computational efficiency, the statistics in Table 2 suggest that for a small number of movable obstacles, for example in the domains SQUARE_1 and SQUARE_2, $m = 2$ may be used. For domains with a larger number of obstacles (ROBOT and RANDOM), $m = 3$ may be employed.

Let us now consider again a simulated Pepper humanoid robot having to move through a cluttered environment as seen in Fig. 1a. Pepper is asked to move to the other room, but its path is blocked by the specific configuration of the tables. Employing the robot model (5) with $m = 2$ and $w^g = 1$, $w^x = 1$, $w^d = 10$, a 7-obstacle solution with a displacement magnitude of 4.4 meters is obtained (see Fig. 15a). Penalizing large displacements by increasing α^o from 10 to 1000, a longer path with a 2-obstacle solution (see Fig. 15b) can be obtained with a lower displacement magnitude of 0.4 meters. We recall here that the cost for the last time step is the deviation of the robot location from its goal.

Baseline comparison. We additionally compare HS4MOD against a NAMO baseline on the SQUARE_2 and ROBOT environments. Similar to [8, 7], the baseline first constructs the robot’s configuration space and computes a path to the goal using an A^* search that minimizes the number of obstacles intersecting the path. The obstacles intersecting the path are then displaced to obtain a collision-free trajectory, following the approach proposed in [20].

Table 1 compares the robot path length, total obstacle displacement, and planning time HS4MOD and the NAMO baseline. In the SQUARE_2 environment, HS4MOD produces a shorter robot path with nearly identical obstacle displacement, while reducing planning time by more than a factor of three. In

Method	<i>SQUARE_2</i>			<i>ROBOT</i>		
	Length (m)	Displacement (m)	Time (s)	Length (m)	Displacement (m)	Time (s)
HS4MOD ($m = 5$)	36.92	7.25	1.36	78.55	9.17	12.23
NAMO Baseline	41.00	7.26	4.57	78.20	10.80	17.82

Table 1: Comparison of robot path length, obstacle displacement and planning time for HS4MOD ($m = 5$) and NAMO baseline.

the ROBOT environment, both methods generate paths of comparable length, but HS4MOD requires less obstacle displacement and achieves a lower planning time. These results demonstrate that HS4MOD, even when $m = 5$ offers a favorable trade-off between solution quality and runtime.

6.2. Nonlinear Robot Model

As argued in Section 4, when a nonlinear robot model is adopted for the optimization problem (3), optimality cannot be guaranteed. We test HS4MOD with the nonlinear models (7) and (8) on *SQUARE_1* and *SQUARE_2*. The corresponding robot trajectories and the obstacle displacements with varying m for both the robot models are visualized in Fig. 13 and Fig. 14, respectively. The results indicate that regardless of the robot model, the trajectories exhibit a similar pattern. However, it is observed that certain obstacles are displaced unnecessarily, such as, when $m = 5$ (last row) in Fig. 13(a), $m = 4$ in Fig. 13(b), and $m = 5$ in Fig. 14(b). This phenomenon can be attributed to the non-convexity of the optimization problem, resulting in solutions that are local optima. Numerical results are presented in Table 3. As argued in Section 4, due to the nonlinearity of the models, $m = 1$ does not guarantee an optimal solution. In contrast to the findings in Table 2, in Table 3, the change in ϵ is not monotonic with an increase in m (note that ϵ is computed with respect to $m = 1$). This behavior results from the interaction between horizon discretization and nonlinear dynamics, whereby increasing the number of slices can alter the feasible trajectory and the associated obstacle displacements without necessarily yielding a progressively improved solution. Therefore, no particular value of m can be identified as providing a consistent trade-off between solution qual-

ity and computational efficiency for the nonlinear models. Moreover, ϵ should be interpreted as a relative measure with respect to the $m = 1$ solution, rather than as a direct measure of the true optimality gap.

Therefore, using a nonlinear model in (3b) precludes any validation on ϵ . Nonetheless, in such scenarios, a linear model can initially be employed to determine the necessary path and displacements using HS4MOD. As a consequence, a trajectory-tracking algorithm can be employed to calculate the requisite controls for the nonlinear model.

6.3. Two Different Problems

We have previously noted how MOD encompasses various classes of motion planning problems. In this Section, we explore two such problems using a linear robot model. Let us consider again the simulated Pepper humanoid robot scenario depicted in Fig.1a, where we remove a table blocking the top doorway. By employing the exact approach, a solution to the classical motion planning problem is obtained in approximately 8.31 seconds (visualized in Fig.16a). This demonstrates that when a collision-free path exists, MOD resolves the classical motion planning problem with all displacements set to zero.

The MCR problem identifies the minimum number of obstacles that must be removed to yield a feasible path. To transform MOD into MCR, we allow collisions with obstacles and modify the cost function (2d) such that it assigns a cost of 1 for collision with an obstacle, and 0 otherwise. The exact approach produces a solution in approximately 6.37 seconds (see Fig. 16b), revealing that at least one obstacle must be removed to achieve a feasible path. MCR, likewise MOD, has been proven to be NP-hard, and HS4MOD may be equally applied to MCR. To evaluate this possibility, we solve the problem using $m = 3$, and obtain a 2-obstacle approximation in 0.67 seconds, as shown in Fig. 16c. This result is interesting in so far as MCR is known to have an $O(\sqrt{n})$ approximation for rectilinear polygons and disks [48]. While the analysis of HS4MOD in the context of MCR is an intriguing avenue for exploration, it falls beyond the scope of this work and is left for future research.

Robot model (5)						Robot model (6)			
	m	CPU [s]	Gap/ ϵ	$J^r[m^2]$	$J^o[m^2]$	CPU [s]	Gap/ ϵ	$J^r[m^2]$	$J^o[m^2]$
SQUARE_1	1	1011.41	0.00	45.12	0.27	1001.34	0.00	47.35	0.35
	2	516.44	0.59	48.02	0.55	11.85	0.45	48.67	0.61
	3	3.58	0.89	44.64	0.72	3.06	1.28	48.49	1.09
	4	2.99	1.97	45.45	1.25	2.58	1.50	49.09	1.22
	5	1.50	2.76	46.21	1.64	1.37	2.83	51.97	1.99
SQUARE_2	1	1601.58	0.00	51.20	0.28	1000.98	0.00	51.22	0.28
	2	27.92	0.62	59.17	0.56	116.37	0.76	59.33	0.65
	3	4.79	0.90	53.87	0.74	7.19	1.06	54.38	0.84
	4	1.32	2.10	59.44	1.35	1.95	3.03	64.91	1.85
	5	1.02	3.08	59.29	1.88	1.36	3.72	65.90	2.22
ROBOT	1	28814.74	0.00	122.99	0.28	28808.67	0.00	132.70	0.32
	2	17367.71	0.25	121.64	0.52	19564.10	0.18	133.00	0.49
	3	128.68	0.37	121.08	0.63	147.85	0.31	127.93	0.66
	4	24.23	0.98	119.32	1.19	28.83	0.71	136.78	0.99
	5	11.62	1.21	142.74	1.28	12.24	0.75	128.16	1.08
RANDOM	1	16009.18	0.00	51.86	1.13	18010.40	0.00	39.15	1.60
	2	12813.12	0.78	49.56	2.24	14413.58	0.77	52.28	2.93
	3	6403.36	0.99	46.68	2.55	771.53	1.35	48.39	3.99
	4	897.14	1.83	48.85	3.71	44.76	1.43	43.72	4.16
	5	14.43	2.05	46.70	4.03	20.47	2.16	47.55	5.44

Table 2: Different statistics for the four domains considered in this paper, while employing the linear robot models in (5) and (6). $m = 1$ corresponds to the scenario with no horizon-slicing. CPU denotes the computation time in seconds for the optimization problem in (3), $J^r = c(\mathbf{p})$, and $J^o = \sum_{i=1}^{\mathcal{M}} \sum_{k=0}^T d_k^i$.

Robot model (7)						Robot model (8)			
	m	CPU [s]	Gap/ ϵ	$J^r[m^2]$	$J^o[m^2]$	CPU [s]	Gap/ ϵ	$J^r[m^2]$	$J^o[m^2]$
SQUARE_1	1	88.59	0.00	52.19	0.20	307.17	0.00	52.31	0.61
	2	96.99	0.09	46.73	0.27	150.38	0.92	41.86	0.24
	3	61.81	0.43	46.89	0.46	128.93	0.46	43.27	0.38
	4	26.53	1.35	47.75	0.89	97.34	0.37	43.83	0.41
	5	12.22	4.69	50.00	2.49	35.24	0.12	42.28	0.57
SQUARE_2	1	141.78	0.00	62.43	0.30	71.53	0.00	67.75	0.65
	2	47.33	0.89	49.75	0.45	29.18	0.44	45.15	0.45
	3	125.09	0.22	52.67	0.85	146.78	0.49	43.76	1.81
	4	86.29	0.61	60.75	3.28	68.66	0.37	45.16	1.42
	5	60.09	0.18	51.55	1.40	74.10	0.72	95.50	3.15

Table 3: Average computation time, Gap- wrt to $m = 1$, path ($J^r = c(\mathbf{p})$) and displacement function ($J^o = \sum_{i=1}^{\mathcal{M}} \sum_{k=0}^T d_k^i$) values while employing the nonlinear robot models in (7) and (8).

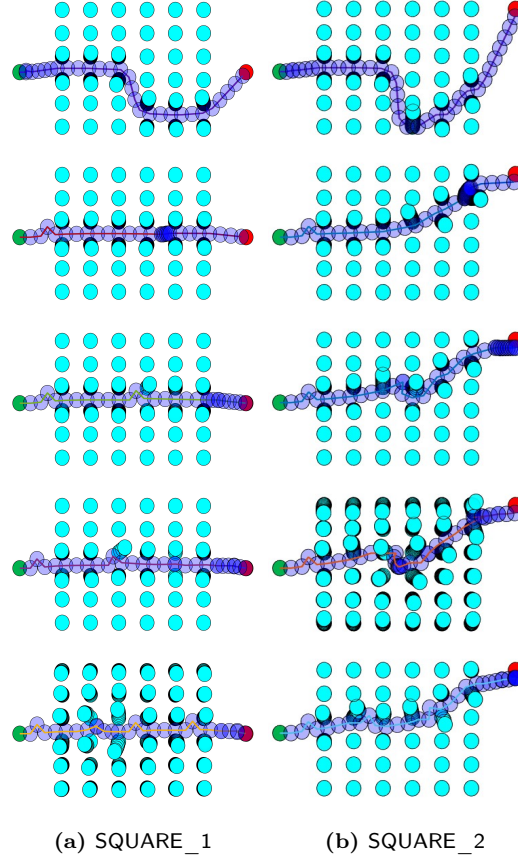


Figure 13: Robot trajectories and obstacle displacements for `SQUARE_1` and `SQUARE_2` domains while the robot moves according to the model in (7). The first row corresponds to $m = 1$ (no horizon slicing), the second row corresponds to $m = 2$, and so on.

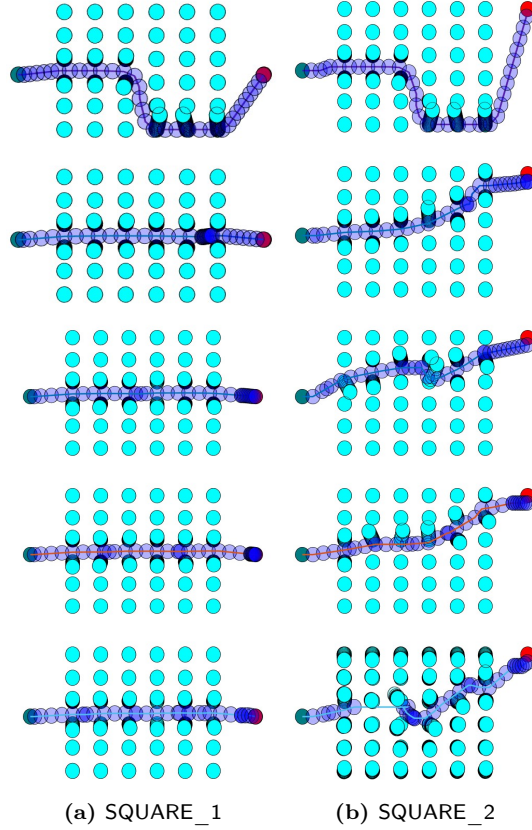


Figure 14: Robot trajectories and obstacle displacements are depicted for the SQUARE_1 and SQUARE_2 domains, where the robot moves according to the model in (8). Each row corresponds to a different value of m : the first row corresponds to $m = 1$ (no horizon slicing), the second row corresponds to $m = 2$, and so on.

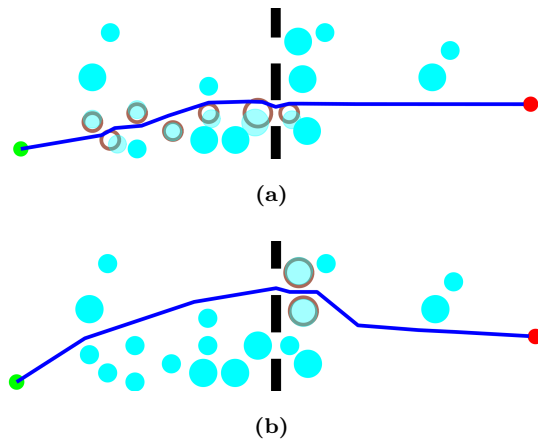


Figure 15: Two trajectories related to the example scenario that was introduced at the beginning of the paper. A humanoid Pepper robot must navigate to the right-hand side of the map, and to that aim it must displace movable obstacles to pass through one of the two available passages. (a) Pepper finds a short path that implies displacing 7 obstacles. (b) Increasing the obstacle displacement factor (J^o) weight from 10 to 1000, the robot finds a path with lower obstacle displacement but with a higher path length.

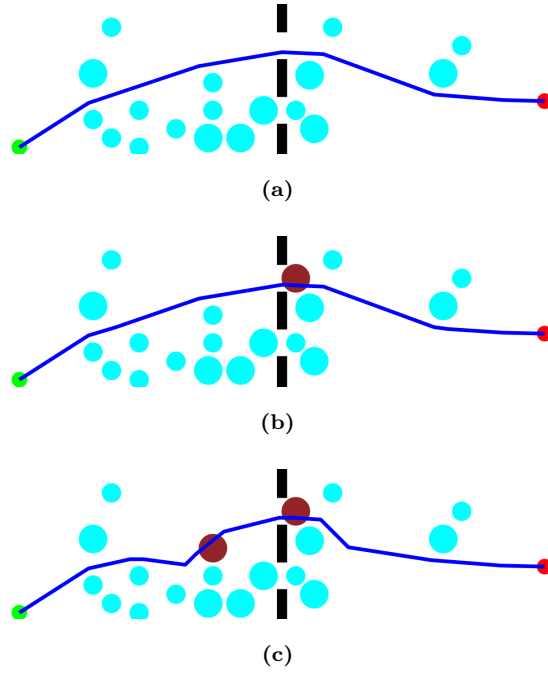


Figure 16: (a) Example of MOD solving the classical motion planning problem when a collision-free path exists. (b) The MOD cost is modified to obtain the MCR path using the exact approach. (c) HS4MOD with $m = 3$ gives an approximate 2-obstacle MCR path.

7. Conclusions and Future Work

This paper formalizes and discusses a *minimum obstacle displacement* (MOD) problem for robot navigation tasks requiring the manipulation of obstacles in the way. This work proves that the problem is NP-hard for polygonal obstacles in the plane, even when such obstacles are modeled as simple line segments. The paper presents an exact approach, which generalizes a number of existing problems in the literature, which is computationally intractable in the worst case. Furthermore, the paper discusses a trade-off between path length and the amount of object displacements, and it introduces an approximate albeit sub-optimal horizon-slicing approach characterized by a high computational gain when compared to the exact approach. Both approaches are validated using different theoretical examples, scenarios with up to 100 obstacles, and different robot motion models. The solution quality of the approximate method is measured in terms of its optimality gap, and the validations suggest that the approximate method can provide a desirable trade-off between the optimality gap and computation time.

The models presented in the paper are characterized by a few limitations. First, we do not deal with obstacles *interacting* with each other, that is, obstacles themselves are not collision-free. Interaction between obstacles can be incorporated by adding collision avoidance constraints among obstacles; similar to (3h) an additional constraint of the form $O(\mathbf{o}_k^j) \cap \bigcup_{i=1, i \neq j}^{|\mathcal{O}|} O(\mathbf{o}_k^i) = \emptyset$ should be added to (3). However, for a large number of obstacles, taking such collisions into account increases the computational burden due to the addition of the corresponding binary variables. Second, the approaches discussed in the paper are restricted to robot and obstacle classes allowing for linearizing the collision constraint since the optimization problem for MOD requires the linearisation of the collision constraint in (3h). For the interaction between circles or axis-aligned rectangles, this can be achieved by introducing four binary constraints (for an example, see (13)-(17)). Other polygonal projections of the robot and the obstacles require a careful formulation of the linear constraints (10)-(11). However,

with an increasing number of obstacles, there is a higher computational burden due to the increased number of auxiliary variables.

As argued throughout the paper, we are not concerned about the weight of the obstacles and assume that the computed displacements can be achieved. Yet, an obstacle might be too heavy for being moved even though it accounts for a minimum displacement solution. This aspect may be incorporated with heavy obstacles being penalized more by assuming a higher weight than other obstacles. Future work includes incorporating additional aspects such as the work done by the robot which addresses similar concerns through the force required to move the obstacles. Furthermore, the proposed method is intended as an offline planning approach, where the required displacement for each obstacle is computed prior to execution under the assumption that the environment map is known. In a practical deployment, however, the robot may have access only to a locally observed portion of the environment through its onboard sensors. In such a setting, HS4MOD could be extended to an online incremental planning framework by repeatedly constructing and solving the horizon-sliced optimization problem using the currently observed environment, and updating the plan as the robot moves and new obstacles or obstacle configurations become available. While the development and evaluation of such an online replanning strategy are beyond the scope of the present work, investigating this extension represents an interesting direction for future research.

We consider environments with movable obstacles, where the robot itself is responsible for displacing the obstacles and therefore the focus is on scenarios where no feasible path initially exists for the robot. Thus, the change in the environment arises from deliberate robot-object interactions, rather than from independently evolving or time-varying dynamics.

One could argue that partitioning the planning horizon based on obstacle encounters would be more effective than using equal-length horizon slices. However, such an approach requires either prior knowledge of the solution trajectory or an additional preprocessing stage to identify the obstacle encounter events, thereby increasing the computational complexity. In contrast, equal horizon

slicing provides a simple, general, and computationally efficient decomposition strategy that can be applied without any prior knowledge of the trajectory. Nevertheless, the investigation of adaptive horizon partitioning strategies represents an interesting direction for future research.

Finally, it is worth noting that the evaluation domains used in this paper have been rather abstract in nature. Yet, this is by no means a limitation, and many practical applications exist to the minimum obstacle displacement problem, such as for example those related to automated logistics [37].

References

- [1] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, Cambridge, UK, 2006.
- [2] F. Mastrogiovanni, A. Sgorbissa, R. Zaccaria, Robust navigation in an unknown environment with minimal sensing and representation, *IEEE Transactions on Systems, Man, and Cybernetics* 39 (1) (2009) 212–229.
- [3] K. Hauser, Minimum constraint displacement motion planning., in: *Proceedings of the 9th International Conference Robotics: Science and Systems (RSS IX)*, Berlin, Germany, 2013.
- [4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, 2009.
- [5] A. Capitanelli, M. Maratea, F. Mastrogiovanni, M. Vallati, On the manipulation of articulated objects in human-robot cooperation scenarios, *Robotics and Autonomous Systems* 109 (2018) 139–155.
- [6] G. Michael, D. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, W.H. Freeman, San Francisco, 1979.
- [7] A. Thomas, F. Mastrogiovanni, Minimum displacement motion planning for movable obstacles, in: *International Conference on Intelligent Autonomous Systems*, Springer, 2022, pp. 155–166.

- [8] M. Stilman, J. J. Kuffner, Navigation among movable obstacles: Real-time reasoning in complex environments, *International Journal of Humanoid Robotics* 2 (4) (2005) 479–503.
- [9] D. Nieuwenhuisen, A. F. van der Stappen, M. H. Overmars, An effective framework for path planning amidst movable obstacles, in: *Proceedings of the 7th Workshop on the Algorithmic Foundations of Robotics (WAFR VII)*, New York, NY, USA, 2008.
- [10] M. Stilman, J. Kuffner, Planning among movable obstacles with artificial constraints, *The International Journal of Robotics Research* 27 (11-12) (2008) 1295–1307.
- [11] J. Van Den Berg, M. Stilman, J. Kuffner, M. Lin, D. Manocha, Path planning among movable obstacles: a probabilistically complete approach, in: *Proceedings of the 8th Workshop on the Algorithmic Foundations of Robotics (WAFR VIII)*, Guanajuato, Mexico, 2009.
- [12] J. J. Weeda, S. Bakker, G. Chen, J. Alonso-Mora, Pushing through clutter with movability awareness of blocking obstacles, in: *2025 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2025, pp. 512–518.
- [13] Y. Zhang, Y. Kantaros, NAMO-LLM: Efficient Navigation Among Movable Obstacles With Large Language Model Guidance, *IEEE Robotics and Automation Letters* 10 (12) (2025) 13026–13033. doi:10.1109/LRA.2025.3621972.
- [14] T. Yang, J. Hwang, J. Jeong, M. Yoon, S.-E. Yoon, Efficient navigation among movable obstacles using a mobile manipulator via hierarchical policy learning, in: *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2025, pp. 17130–17137.
- [15] G. Wilfong, Motion planning in the presence of movable obstacles, *Annals of Mathematics and Artificial Intelligence* 3 (1) (1991) 131–150.

- [16] J. Basch, L. Guibas, D. Hsu, A. T. Nguyen, Disconnection proofs for motion planning, in: Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA), Seoul, Korea, 2001.
- [17] L. Zhang, Y. J. Kim, D. Manocha, A simple path non-existence algorithm using C-obstacle query, in: Proceedings of the 7th Workshop on the Algorithmic Foundations of Robotics (WAFR VII), New York, NY, USA, 2008.
- [18] S. Li, N. T. Dantam, Learning proofs of motion planning infeasibility, in: Proceedings of the 17th International Conference Robotics: Science and Systems (RSS XVII), Anywhere on Earth, 2021.
- [19] S. Li, N. T. Dantam, Scaling infeasibility proofs via concurrent, codimension-one, locally-updated coxeter triangulation, IEEE Robotics and Automation Letters 8 (12) (2023) 8303–8310.
- [20] A. Thomas, F. Mastrogiovanni, M. Baglietto, Locally optimal solutions to constraint displacement problems via path-obstacle overlaps, Robotics and Autonomous Systems 196 (2026) 105251. doi:<https://doi.org/10.1016/j.robot.2025.105251>.
URL <https://www.sciencedirect.com/science/article/pii/S0921889025003483>
- [21] M. Stilman, J.-U. Schamburek, J. Kuffner, T. Asfour, Manipulation planning among movable obstacles, in: Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy, 2007.
- [22] M. Dogar, S. Srinivasa, A framework for push-grasping in clutter, in: Proceedings of the 7th International Conference Robotics: Science and Systems (RSS VII), Los Angeles, CA, USA, 2011.
- [23] A. Krontiris, K. E. Bekris, Dealing with difficult instances of object rearrangement, in: Proceedings of the 11th International Conference Robotics: Science and Systems (RSS XI), Rome, Italy, 2015.

- [24] H. Karami, A. Thomas, F. Mastrogiovanni, Task allocation for multi-robot task and motion planning: a case for object picking in cluttered workspaces, in: Proceedings of the 20th International Conference of the Italian Association for Artificial Intelligence (AIXIA), Milan, Italy, 2020.
- [25] L. P. Kaelbling, T. Lozano-Pérez, Integrated task and motion planning in belief space, *The International Journal of Robotics Research* 32 (9-10) (2013) 1194–1227.
- [26] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, P. Abbeel, Combined task and motion planning through an extensible planner-independent interface layer, in: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014.
- [27] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, L. E. Kavraki, Incremental task and motion planning: a constraint-based approach, in: Proceedings of the 12th International Conference Robotics: Science and Systems (RSS XII), Ann Arbor, Michigan, USA, 2016.
- [28] C. R. Garrett, T. Lozano-Perez, L. P. Kaelbling, FFRob: leveraging symbolic planning for efficient task and motion planning, *The International Journal of Robotics Research* 37 (1) (2018) 104–136.
- [29] A. Thomas, F. Mastrogiovanni, M. Baglietto, MPTP: motion-planning-aware task planning for navigation in belief space, *Robotics and Autonomous Systems* 141 (2021) 103786.
- [30] H. Karami, A. Thomas, F. Mastrogiovanni, A task and motion planning framework using iteratively deepened and/or graph networks, *Robotics and Autonomous Systems* 189 (2025) 104943. doi:<https://doi.org/10.1016/j.robot.2025.104943>.
URL <https://www.sciencedirect.com/science/article/pii/S0921889025000296>

- [31] K. Hauser, The minimum constraint removal problem with three robotics applications, *The International Journal of Robotics Research* 33 (1) (2014) 5–17.
- [32] L. I. R. Castro, P. Chaudhari, J. Tůmová, S. Karaman, E. Frazzoli, D. Rus, Incremental sampling-based algorithm for minimum-violation motion planning, in: *Proceedings of the 52nd IEEE Conference on Decision and Control (CDC)*, Florence, Italy, 2013.
- [33] B. Xu, H. Min, Solving minimum constraint removal (mcr) problem using a social-force-model-based ant colony algorithm, *Applied Soft Computing* 43 (2016) 553–560.
- [34] A. Krontiris, K. E. Bekris, Trade-off in the computation of minimum constraint removal paths for manipulation planning, *Advanced Robotics* 31 (23-24) (2017) 1313–1324.
- [35] B. Xu, L. Chen, K. Xu, Deep learning algorithm for minimum constraint removal (MCR) problem, in: *Proceedings of the 12th International Conference on Machine Learning and Computing (ICMLC)*, Shenzhen, China, 2020.
- [36] A. Thomas, F. Mastrogiovanni, M. Baglietto, Revisiting the Minimum Constraint Removal Problem in Mobile Robotics, in: *Intelligent Autonomous Systems 18*, Springer Nature Switzerland, Cham, 2024, pp. 31–41.
- [37] I. S. Mohamed, A. Capitanelli, F. Mastrogiovanni, S. Rovetta, R. Zaccaria, Detection, localisation, and tracking of pallets using machine learning techniques and 2D range data, *Neural Computing and Applications* 32 (2019) 8811–8828.
- [38] H. Kellerer, U. Pferschy, D. Pisinger, H. Kellerer, U. Pferschy, D. Pisinger, The multiple-choice knapsack problem, *Knapsack Problems* (2004) 317–347.

- [39] J. Van Den Berg, P. Abbeel, K. Goldberg, LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information, *The International Journal of Robotics Research* 30 (7) (2011) 895–913.
- [40] J. Van Den Berg, S. Patil, R. Alterovitz, Motion planning under uncertainty using iterative local optimization in belief space, *The International Journal of Robotics Research* 31 (11) (2012) 1263–1278.
- [41] C. Park, J. S. Park, D. Manocha, Fast and bounded probabilistic collision detection for high-DOF trajectory planning in dynamic environments, *IEEE Transactions on Automation Science and Engineering* 15 (3) (2018) 980–991.
- [42] A. Thomas, F. Mastrogiovanni, M. Baglietto, Safe motion planning with environment uncertainty, *Robotics and Autonomous Systems* 156 (2022) 104203. doi:<https://doi.org/10.1016/j.robot.2022.104203>.
URL <https://www.sciencedirect.com/science/article/pii/S0921889022001166>
- [43] R. Salcedo, Solving nonconvex nonlinear programming and mixed-integer nonlinear programming problems with adaptive random search, *Industrial & Engineering Chemistry Research* 31 (1) (1992) 262–273.
- [44] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, A. Mahajan, Mixed-integer nonlinear optimization, *Acta Numerica* 22 (2013) 1–131.
- [45] V. V. Naik, Mixed-integer quadratic programming algorithms for embedded control and estimation, Ph.D. thesis, IMT School for Advanced Studies Lucca (2018).
- [46] J. Löfberg, YALMIP: a toolbox for modeling and optimization in MATLAB, in: *Proceedings of the IEEE International Symposium on Computer-Aided Control Systems Design (CACSD)*, Taipei, Taiwan, 2004.

- [47] A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, *Mathematical programming* 106 (2006) 25–57.
- [48] S. Bandyapadhyay, N. Kumar, S. Suri, K. Varadarajan, Improved approximation bounds for the minimum constraint removal problem, *Computational Geometry* 90 (2020) 101650.