2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Computational Tradeoff in Minimum Obstacle Displacement Planning for Robot Navigation

Antony Thomas and Giulio Ferro and Fulvio Mastrogiovanni and Michela Robba

***Abstract*— In this paper, we look into the minimum obstacle displacement (MOD) planning problem from a mobile robot motion planning perspective. This problem finds an optimal path to goal by displacing movable obstacles when no path exists due to collision with obstacles. However this problem is computationally expensive and grows exponentially in the size of number of movable obstacles. This work looks into approximate solutions that are computationally less intensive and differ from the optimal solution by a factor of the optimal cost.**

***Index Terms*— Collision Avoidance, Constrained Motion Planning, Obstacle displacement planning**

I. Introduction

Classical robot motion planning approaches search for a feasible path from start to goal. Feasibility is often application specific and involve different constraints such as obstacle avoidance, mechanical limits, actuator limits, state uncertainties. A complete planner will search for all possible paths and if no feasible path exist due to constraint violation, the planner terminates reporting failure. However, in certain application domains it may be possible to alter some of the constraints to produce a feasible path. For example, manipulator robots often rearrange or move obstacles aside to complete a task, humanoid robots may need to move aside obstacles to reach a goal— reposition chairs or open doors. When environmental and state uncertainties are represented by particles, finding the path that minimizes the probability of collision can be formulated as finding a path that collides with minimum number of particles [1].

Consider a humanoid robot in an office or home environment. While navigating through clutter we would envision the humanoid to exhibit human like navigation. A humanoid with such a capability will not always avoid obstacles but often rearrange or reposition movable obstacles such as coffee tables or chairs so as to synthesize a feasible path. Though there are different notions of optimality, we would want the humanoid to expend the minimum amount of *work* while moving the obstacles. This can depend on a number of factors such as, the maximum forces available at the obstacle contact point, displacement of the contact point, number of obstacles moved. In this paper, we focus on the displacement factor, that is, *how can a robot reach its goal by displacing obstacles while minimizing the total obstacle displacements* (overall displacement magnitudes). This also has applications in search and rescue wherein the robot minimizes debris displacement to reach the target quickly.

*Department of Informatics, Bioengineering, Robotics, and Systems Engineering, University of Genoa, Via All'Opera Pia 13, 16145 Genoa, Italy. *antony.thomas@dibris.unige.it, giulio.ferro@unige.it, fulvio.mastrogiovanni@unige.it, michela.robba@unige.it*

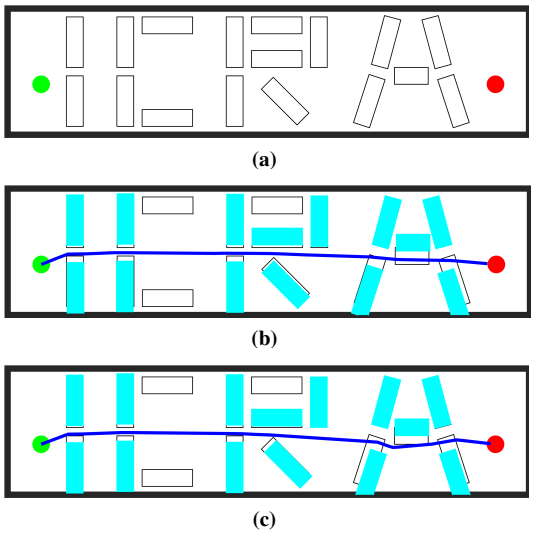


Fig. 1: (a) A robot asked to move from left to right in the ICRA domain with 17 movable obstacles (only translation allowed). (b) After 321 s, optimal solution is obtained with displaced obstacles shown in cyan. (c) An 0.24-optimal solution (details on ϵ-optimality in Section V) obtained in 25 s.

In this paper, we formulate a Minimum Obstacle Displacement (MOD) problem that minimizes a weighted cost of robot path length and total obstacle displacement magnitudes as the robot navigates to its goal. This problem is NP-hard and therefore scenarios with even small number of obstacles can be computationally intensive (see Fig. 1). Thus, it is appropriate to search for approximate methods and this work looks into one such method by dividing an MOD problem into sub-problems. This division into sub-problems is an approximation as MOD does not exhibit the optimal substructure property [2], that is, optimal solutions to sub-problems are not necessarily optimal. Consider a simple scenario shown in Fig. 2. The robot is asked to move from left (green in figure) to right (red in figure) in the presence of movable obstacles (A, B and C in figure). The robot start and goal locations are chosen such that the robot path length for both the upper (displacing A and B) and the lower (displacing C) paths are equal in magnitude. In Fig. 2b, the problem is split into two sub-problems. Since minimum displacement is solicited (path lengths equal as argued above), the robot chooses to move A by 2 units. For the next sub-problem, robot moves B by 2 units to reach the goal. Thus, the total displacement magnitude is 4 units. On the other hand, displacing C by 3 units clears a feasible path

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10161372

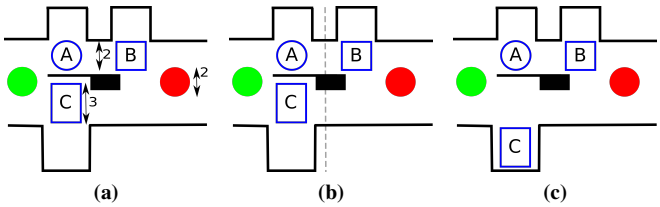


Fig. 2: (a) A simple domain where the robot is asked to move from left to right in the presence of three movable obstacles. (b) Solving the MOD problem considering two sub-problems (left and right of dashed line). This leads to a feasible path by moving obstacles A and B with total displacement magnitude of 4 units. (c) The optimal solution is obtained by displacing obstacle C by 3 units.

for the robot as shown in Fig. 2c and is the optimal solution. The main contributions of the paper are: 1) Formulation and computation an of the optimal solution to the MOD problem, albeit computationally intensive, 2) Computation of approximate ϵ-optimal solutions, that is, solutions that differ from the optimal solution by a factor of ϵ. ϵ-optimal solutions provide a very reasonable approximation with significant computational efficiency. Finally, in Section V, we compute an upper bound for ϵ.

In this work, we are not concerned about how to move the obstacles, that is, we ignore the weight, size of the obstacles and assume that the computed displacements can be achieved by robot-obstacle or human-obstacle interaction. From an algorithmic perspective, this allows to perform the implementation on 2D projections of the 3D environment.

II. Related Work

This problem discussed herein is closely related to Navigation Among Movable Obstacles (NAMO) in [3]–[5]. NAMO class of problems are shown to be NP-hard if the final locations of the obstacles are unspecified, and PSPACE-hard when the locations are specified [6]. Most approaches thus solve a subclass of problems, focusing on efficiency. Manipulation among clutter or rearrangement planning in clutter [7]–[10] is another related class of problems wherein obstacles may need to be displaced to pick a target of interest. The Minimum Constraint Removal problem (MCR) [1] finds the minimum number of obstacles/constraints to be displaced to obtain a feasible path. Different algorithms for MCR exist in the literature [11]–[14] and is proven to be NP-hard for convex polygonal obstacles [1]. Approaches for integrated task and motion planning [15]–[19] also bears resemblance to the problem discussed in this paper since task execution may require removing (or adding) constraints at the motion planning level. Disconnection proving in motion planning [20]–[22] also pose related challenges since these approaches try to find constraints that prevents robot motion to goal.

The work in [23] examine minimum displacement planning for movable obstacle through a 2 stage process. The first stage proceeds by finding a path through movable obstacle while minimizing robot-obstacle overlaps. The overlapping obstacles are then displaced iteratively by a distance factor until no overlap. Yet, the solutions returned are not necessarily optimal and no comment is made on the quality of the same. Probably the work that is more close to our approach is the Minimum Constraint Displacement (MCD) problem introduced by Hauser in [24]. The approach in [24] proceeds by (1) building a probabilistic roadmap of robot configurations and (2) sampling random obstacle displacements. The solution quality is improved by iteratively expanding the roadmap and the displacement samples. The method is shown to asymptotically approach the true optimum (which is unknown) with increasing robot configurations in the roadmap and with growing displacement samples. In contrast, we compute approximate solutions that deviates from the optimal solution cost by a factor of ϵ. The approximate solutions offer significant computational advantage over the optimal solution. Further, MCD focus 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. We plan a trajectory taking into account the robot dynamics and other constraints such as control limits, obstacle avoidance, to find the sequence of control actions that moves the robot to the goal.

III. Problem Definition

Throughout this paper we shall denote vectors by bold lower case letters, that is $\mathbf{x}$ and its components by lower case letters. The transpose of $\mathbf{x}$ will be denoted by $\mathbf{x}^T$ and its Euclidean norm by $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$. Sets will be denoted using mathcal fonts, that is, $\mathcal{S}$ or enclosed within braces ($\{\cdot\}$) and its cardinality will be denoted by $|\mathcal{S}|$.

Let $\mathbf{x}_k$ denote the robot state at any time k. For instance, in mobile robot navigation $\mathbf{x}_k$ may denote the robot pose at time k. We consider a standard motion model for the robot given by $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$, where $\mathbf{u}_k$ is the applied control action at time k. Further, let $\mathbf{x}^s$ denote the start state and $\mathbf{x}^g$ denote the goal state. Let $\mathcal{O} = \{\mathbf{o}^i \mid 1 \leq i \leq |\mathcal{O}|\}$ denote the set of movable obstacle in the environment. By abuse of notation we will use $\mathbf{o}^i$ to denote both the i-th obstacle as well as its state. The obstacles are associated with a displacement set $\mathcal{D} = \{\mathbf{d}^i \in \mathbb{R}^n \mid 1 \leq i \leq |\mathcal{O}|\}$ that represents the corresponding obstacle displacements. The new obstacle location after being displaced by $\mathbf{d}^i$ will be denoted by $\mathbf{o}^i(\mathbf{d}^i)$. Since the obstacles can either rotate or translate or perform both, $\mathbf{d}^i$ belongs to a displacement space of arbitrary dimension. We can now define the MOD planning problem.

Definition 1. The Minimum Obstacle Displacement (MOD) planning problem finds a sequence of control actions $\mathbf{u}_1, \ldots, \mathbf{u}_{T-1}$ that navigates the robot from $\mathbf{x}^s$ to $\mathbf{x}^g$ with obstacle displacements $\mathbf{d}^1, \ldots, \mathbf{d}^n$ while minimizing the cost $C(\mathbf{x}, \mathcal{D})$ given by

$$C = w^x c(\mathbf{x}) + w^d \sum_{i=1}^{|\mathcal{O}|} c^i(\mathbf{d}^i) \tag{1}$$

such that 1) $\mathbf{x}_1 = \mathbf{x}^s$, 2) $\mathbf{x}_T = \mathbf{x}^g$, and 3) $\mathbf{x}_k \cap \bigcup_{i=1}^{n} \mathbf{o}^i(\mathbf{d}^i) = \{\emptyset\}$, $\forall k \in \{1, \ldots, T\}$.

In (1), $c(\mathbf{x})$ is a function of robot location (say, path length), $c^i(\mathbf{d}^i)$ is a function of displacement magnitudes and w^x, w^d are the respective weights. Constraints 1 and 2 satisfies the endpoint constraints. The last condition guarantees that the robot does not intersect with the displaced obstacles.

A straightforward but exhaustive approach is to consider all the $2^{|\mathcal{O}|}$ subsets of the movable obstacles with different displacements for each subset and finally selecting the minimum from among them. Note that the subsets for which there exists a solution (not necessarily optimal) are also the different set of obstacles such that removing them from the workspace connects the start and the goal. In fact, by setting $w^x = 0$ and restricting d^i to two values, $d^i = 0$ for obstacle being present and $d^i = 1$ for obstacle removed, MOD solves the MCR problem which is proved to be NP-hard [1]. Therefore, by reduction, MOD is NP-hard.

IV. The Minimum Obstacle Displacement Problem

In this section we present an exact algorithm for MOD which gives optimal solutions, albeit computationally expensive as the set $\mathcal{O}$ and the robot workspace grows.

A. Modeling Obstacle Displacement

As noted before, in this work we are not concerned about how to move the obstacles and assume that obstacles can be displaced irrespective of the nature of the robot-obstacle interaction required. Thus, the 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 and a rotation (the superscript i will be often dropped to avoid clutter). This gives us the following linear model

$$\begin{aligned} o^x_{k+1} &= o^x_k + \tau s^x_k \\ o^y_{k+1} &= o^y_k + \tau s^y_k \\ o^\theta_{k+1} &= o^\theta_k + \tau s^\theta_k \end{aligned} \tag{2}$$

where τ is the duration of a time-step and $\mathbf{s}^i_k = [s^{i,x}_k, s^{i,y}_k, s^{i,\theta}_k]$ specify the velocities. Thus, $[\tau s^x_k, \tau s^y_k, \tau s^\theta_k]$ is the displacement between two time-steps for the i-th obstacle and the overall displacement $\mathbf{d}^i = [\sum_{k=1}^{T-1} \tau s^x, \sum_{k=1}^{T-1} \tau s^y, \sum_{k=1}^{T-1} \tau s^\theta]$. We may write (2) compactly as $\mathbf{o}_{k+1} = g(\mathbf{o}_k, \mathbf{s}_k)$.

B. Robot Model

In general we may consider any standard motion model for the robot. In this paper, we consider two different linear models. The first is a trivial model similar to the obstacle model in (2)

$$\begin{aligned} x_{k+1} &= x_k + \tau u^x_k \\ y_{k+1} &= y_k + \tau u^y_k \\ \theta_{k+1} &= \theta_k + \tau u^\theta_k \end{aligned} \tag{3}$$

where $\mathbf{x}_k = [x_k, y_k, \theta_k]$ is the robot pose at time k and $\mathbf{u}_k = [u^x_k, u^y_k, u^\theta_k]$ is the applied control. We also consider the linear dynamics given in [25] with state $\mathbf{x}_k = [x_k, y_k, v^x_k, v^y_k]$ consisting of its location and velocity with acceleration input $\mathbf{u}_k = [a^x_k, a^y_k]$. The corresponding model is

$$\begin{aligned} x_{k+1} &= x_k + \tau v^x_k + (\tau^2/2) a^x_k \\ y_{k+1} &= y_k + \tau v^y_k + (\tau^2/2) a^y_k \\ v^x_{k+1} &= v^x_k + \tau a^x_k \\ v^y_{k+1} &= v^y_k + \tau a^y_k \end{aligned} \tag{4}$$

where τ is the duration of a time step. The use of linear models allow to obtain global optimal solutions, albeit dependent on the nature of the objective function and other constraints.

C. The optimization problem

We formulate the MOD problem as an optimization problem that finds a feasible path for the robot, minimizing the path length and obstacle displacement magnitudes. While doing so, the control inputs and state variables must lie within their respective feasible sets and the robot should not collide with the obstacles. The overall optimization problem is thus formalized as

$$\min \quad \alpha^r J^r + \alpha^o J^o \tag{5a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{5b}$$

$$\mathbf{o}_{k+1} = g(\mathbf{o}_k, \mathbf{s}_k) \tag{5c}$$

$$\underline{\mathbf{x}} \le \mathbf{x}_k \le \overline{\mathbf{x}} \tag{5d}$$

$$\underline{\mathbf{u}} \le \mathbf{u}_k \le \overline{\mathbf{u}} \tag{5e}$$

$$\underline{\mathbf{o}} \le \mathbf{o}_k \le \overline{\mathbf{o}} \tag{5f}$$

$$\underline{\mathbf{s}} \le \mathbf{s}_k \le \overline{\mathbf{s}} \tag{5g}$$

$$\left\| \mathbf{x}_k - \mathbf{o}^i_k \right\| \ge r^r + r^o \tag{5h}$$

$$\forall k \in \{1, \ldots, T-1\} \text{ and } \forall i \in \{1, \ldots, |\mathcal{O}|\}$$

where

$$J^r = \sum_{k=1}^{T-1} \left\| \mathbf{x}_{k+1} - \mathbf{x}_k \right\|^2 \; + \; \alpha^{re} \left\| \mathbf{x}_T - \mathbf{x}^g \right\|^2 \tag{6}$$

the first term minimizes the robot path length and the second term drives the robot to the goal with α^{re} being its weight and

$$J^o = \sum_{k=1}^{T-1} \sum_{i=1}^{|\mathcal{O}|} \; \tau \left\| \mathbf{s}^i_k \right\|^2 \tag{7}$$

measures the obstacle displacement magnitudes. The weights α^r, α^o result in relative trade-off between longer robot paths and larger obstacle displacements. The robot and obstacle dynamics are followed in constraints (5b)-(5c) subject to the upper and lower bounds on the state and control variables as given in (5d)-(5g). Finally (5h) represents the collision avoidance constraint between the robot and the obstacles with r^r, r^o denoting the robot and obstacle radii, respectively. The endpoint conditions, that is, $\mathbf{x}_1 = \mathbf{x}^s$ and $\mathbf{x}_T = \mathbf{x}^g$ are omitted to avoid further clutter. We note here that in Definition 1, $c(\mathbf{x})$ and $c^i(\mathbf{d}^I)$ where defined to be general functions of robot location and obstacle displacement, respectively. Comparing the above formulation with (1) in Definition 1, it is easily identified that, $\alpha^r = w^x$, $\alpha^o = w^d$, $J^r = c(\mathbf{x})$ and $J^o = \sum_{i=1}^{|\mathcal{O}|} c^i(\mathbf{d}^i)$. For brevity, the objective function (5a) will be denoted using $J = \alpha^r J^r + \alpha^o J^o$.

D. Collision Constraint

At this point, it is essential to spend a few words regarding the modeling of the collision avoidance constraint in (5h). When the robot model (5b) is linear the optimization problem (5) is a non-convex nonlinear program (NLP). However, the only element of non-convexity is the collision avoidance constraint in (5h) (all the other constraints are linear). We thus linearize the collision avoidance constraint 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 step k. The constraint (5h) is thus rewritten as

$$x_k - o_k^{i,x} \geq r^r + r^o - \xi_k^{1,i} M \tag{8}$$

$$-\left(x_k - o_k^{i,x}\right) \geq r^r + r^o - \xi_k^{2,i} M \tag{9}$$

$$y_k - o_k^{i,y} \geq r^r + r^o - \xi_k^{3,i} M \tag{10}$$

$$-\left(y_k - o_k^{i,y}\right) \geq r^r + r^o - \xi_k^{4,i} M \tag{11}$$

$$\sum_{j=1}^{4} \xi_k^{j,i} \leq 3 \tag{12}$$

$\forall k \in \{1,\ldots,T-1\}$ and $\forall i \in \{1,\ldots,|\mathcal{O}|\}$, where M is an appropriate large constant. Thus by rewriting the single constraint (5h) into linear constraints (8)-(12), the optimization problem in (5) is converted into an Mixed Integer Quadratic Program (MIQP). MIQP is 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 that guarantee global optimality for this type of optimization problems. Nonlinear and non-convex robot models provide weak theoretical guarantees and the optimization often results in local minima. Thus, in such settings, a linear model may be employed first to find the required path. A trajectory tracking algorithm can then be used to compute the controls for the nonlinear model.

V. Horizon Slicing

So far we have formulated the MOD problem as an optimization problem (5). In MIQP problems the worst-case computational complexity grows exponentially with the number of binary/integer decision variables [26]. We note here that for each obstacle we have introduced four binary variables to formulate the optimization as an MIQP problem. The number of binary variables is thus $2^{4\times T\times|\mathcal{O}|}$, where T is the number of time steps required to reach the goal. Thus, with increase in the number of movable obstacles and the environment size (larger the environment, larger is the T) the optimization tends to be computationally infeasible. The computational burden can be reduced by decreasing the number of variables. One way to achieve this is by solving subsets of the problem, each of which is computationally inexpensive compared to the solving the complete optimization problem at once. To reduce the computational burden (at the cost of optimality) we propose splitting the optimization problem into m sub-problems, each of duration T' such that $mT' = T$. We slice the number of time steps required to reach the goal into smaller duration and solve the optimization problem for each slice. Once each sub-problem is optimized, the values of time dependent variables are passed on to the following sub-problem.

Let the optimal objective function without horizon slicing be denoted by $J^\star$. Let us also denote by c_k, the cost at each time step. $J^\star$ can thus be written as

$$\begin{aligned} J^\star &= \sum_{k=1}^{T} c_k^\star = \sum_{k=1}^{T'} c_k^\star + \ldots + \sum_{k=(m-1)T'+1}^{T} c_k^\star \\ &= J^\star_{1\to T'} + \ldots + J^\star_{(m-1)T'+1\to T} = \sum_{i=1}^{m} J^\star_{(i-1)T'+1\to iT'} \end{aligned} \tag{13}$$

where $i \in \{1,\ldots,m\}$. Now let us consider the case where we slice the horizon and solve m different optimization problems. The overall objective function J^s is obtained by adding the individual objectives

$$J^s = \sum_{k=1}^{T} c_k = \sum_{i=1}^{m} J^s_{(i-1)T'+1\to iT'} \tag{14}$$

where $J^s_{(i-1)T'+1\to iT'}$ represent the objective function of the $i-$the sub-problem. As noted before, from an algorithmic perspective, a subset of an optimal solution is not necessarily an optimal solution. However, it is easily verified that the solutions to all the subsets is naturally a feasible solution. Thus, the horizon slicing approach results in a feasible solution to the original problem and therefore the value of J^s must be an upper bound, that is, $J^s \geq J^\star$. Since J^s is an upper bound, we have

$$\begin{aligned} J^s &= \sum_{i=1}^{m} J^s_{(i-1)T'+1\to iT'} = \sum_{i=1}^{m} \left(J^\star_{(i-1)T'+1\to iT'} + \delta^i\right) \\ &\leq \left(\sum_{i=1}^{m} J^\star_{(i-1)T'+1\to iT'}\right) + m\delta^{max} \leq J^\star + m\delta^{max} \end{aligned} \tag{15}$$

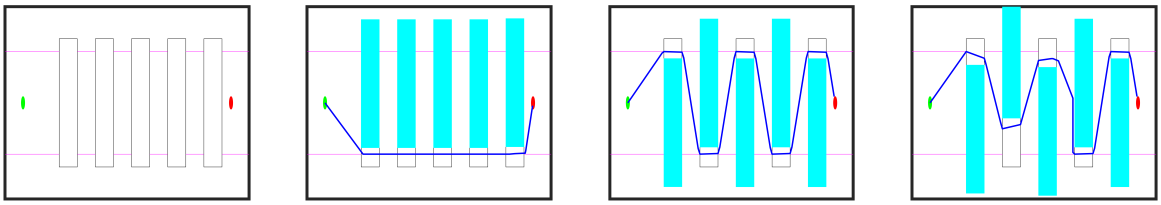
noindent where $\delta^i = J^s_{(i-1)T'+1\to iT'} - J^\star_{(i-1)T'+1\to iT'}$ and $\delta^{max} = \max\{\delta^i \mid 1 \leq i \leq m\}$. From (15), we thus have

$$J^\star \leq J^s \leq J^\star + m\delta^{max} \tag{16}$$

Let the deviation from the optimal solution $J^s - J^\star$ be a factor of $J^\star$, that is, $J^s - J^\star = \epsilon J^\star$. Ideally, we would want $\epsilon \to 0$ with considerable computational efficiency. In the optimization literature the standard metric used to compute the relative distance between an approximate solution and the optimal solution is the *optimality gap*. ϵ essentially computes this relative distance as shown below

$$\text{optimality gap} = \frac{J^s - J^\star}{J^\star} = \frac{\epsilon J^\star}{J^\star} = \epsilon \tag{17}$$

From (16), we have $J^s - J^\star \leq m\delta^{max} \implies \epsilon J^\star \leq m\delta^{max}$ and therefore $\epsilon \leq m\delta^{max}/J^\star$. Clearly, for a given problem as the number of horizon slices or sub-problems increase, the value of ϵ also grows. The exact value is problem specific (see Fig. 3) since it depends on a number of parameters such as the robot and obstacle dynamics, size of the environment, distribution of the obstacles in the environment. Yet, it is easily verified that horizon slicing computes a $m-$approximation as $\delta_{max}/J^\star < 1$ since δ^{max} is the maximum deviation of a sub-problem from its corresponding



(a) (b) (c) (d)

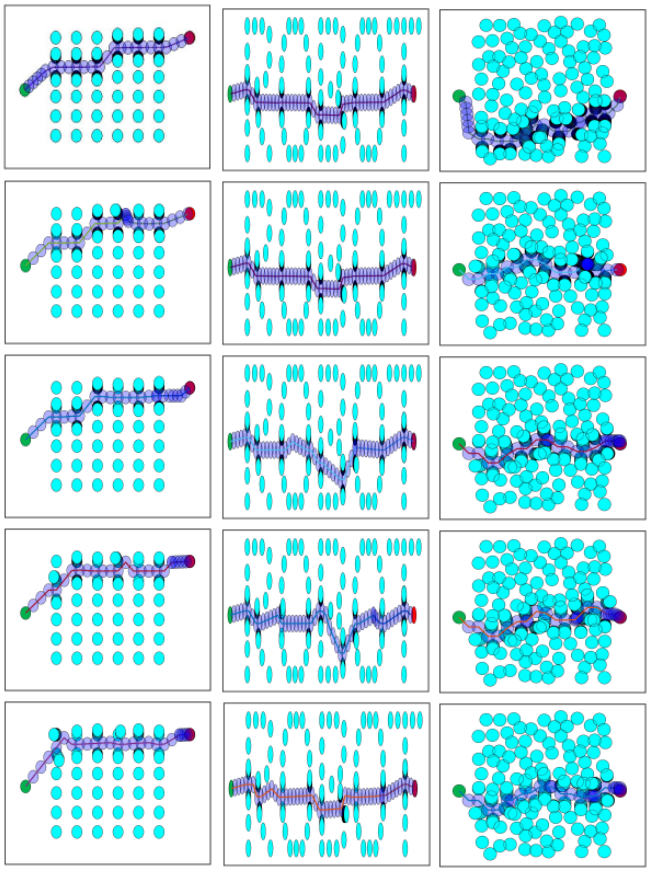
Fig. 3: A robot asked to move from left to right in the presence of 5 movable obstacles. The y-coordinate of the robot is constrained to lie within the magenta lines. (a) Obstacles can only translate in the vertical direction. $m = 1$ and $m = 2$ produced similar trajectories (obstacle displacements shown in cyan). (c)-(d) Odd obstacles (counting from left) can only move downward and even obstacles can only move upward. Optimal solution is given in (c). $m = 2$ gives a sub-optimal solution as shown in (d) with the first sub-problem involving the first 3 obstacles (evident from the vertical trajectory after the third obstacle).

optimal solution, giving $\epsilon < m$. In Section VI, we provide an empirical analysis to determine the best m by performing different experiments while varying some of the parameters stated above.

VI. Evaluation

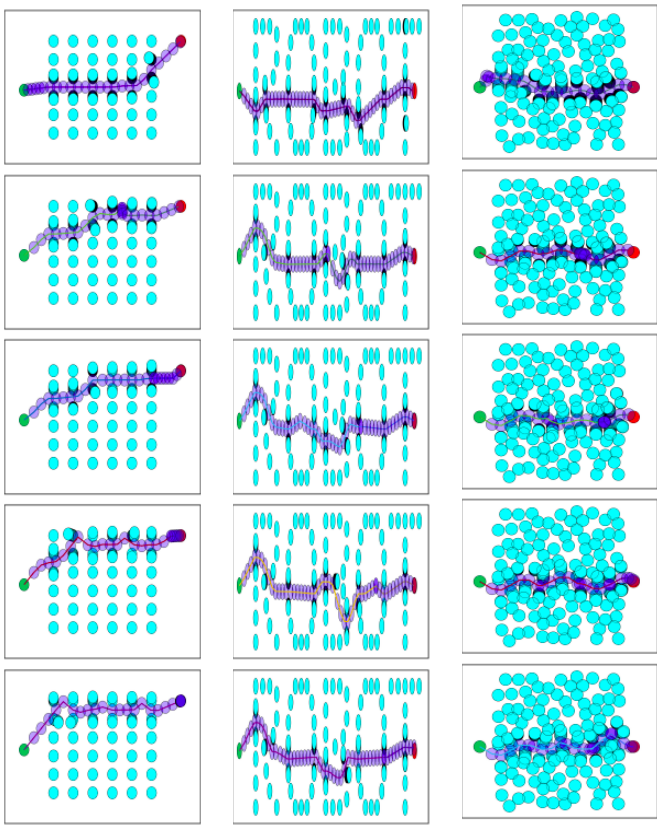
In this section we evaluate our approach using (1) the naive approach that is computationally expensive and (2) horizon slicing. The Optimization problem in (5) is performed using the IBM ILOG CPLEX Optimization Studio V12.10.0 in MATLAB under YALMIP interface [27]. Our method is tested on three different domains, namely, (1) *SQUARE* domain with 36 obstacles, (2) *ROBOT* domain with 74 obstacles, and (3) *RANDOM* domain with 100 obstacles. As discussed in Section IV, m denotes the number of horizon slices and $m = 1$ correspond to the naive approach without the horizon slicing. Currently, the values of α^{re}, α^{r} and α^{o} are chosen via empirical tuning and for all the experiments we use $\alpha^{re} = 10$, $\alpha^{r} = 0.5$ and $\alpha^{o} = 100$. The performance is evaluated on an Intel® i7-10850H CPU @ 2.70GHz with 32 GB RAM under Windows 11.

We first test our method employing the robot dynamics given in (3). The robot trajectory and the obstacle displacements computed are visualized in Fig. 4. Various statistics for the same are presented in Table I under the Robot model (3) column. The naive optimization (without horizon slicing) is denoted by $m = 1$ and is computationally expensive as seen from Table I. As it has been argued before, the computational complexity is exponential in the number of binary variables (proportional to the number of movable obstacles) and is evident from the computational time in *ROBOT* and *RANDOM* domains, with 74 and 100 obstacles, respectively. Additionally, the complexity also depends on (1) the spread of the obstacles with respect to the start and goal location of the robot and (2) the size of the environment. These two factors result in higher computation time for the *ROBOT* domain (79 m×18 m) with 74 obstacles than the *RANDOM* domain (24 m×24 m) with 100 obstacles. Analogous results are seen when employing the linear robot model in (4)— the robot trajectory and obstacle movements visualized in Fig. 5. Clearly, in all the domains, for both the linear robot models, as m increases, the deviation from the optimal solution increases resulting in sub-optimal solutions as corroborated by the rise in ϵ values. Yet, higher values of m results in great computational efficiency. Keeping in mind a pragmatic tradeoff between optimality and computational efficiency, the statistics in Table I suggest that for small number of movable obstacles, $m = 2$ may be used and for a larger number, $m = 3$ may be employed.



(a) *SQUARE* (b) *ROBOT* (c) *RANDOM*

Fig. 4: Robot trajectory and obstacle displacements while the robot moves from left to right under model (3). Figures correspond to $m = 1, \ldots, 5$ starting from the first row.

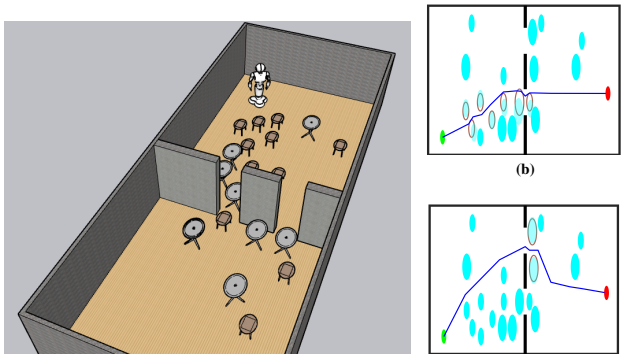


(a) *SQUARE* **(b)** *ROBOT* **(c)** *RANDOM*

Fig. 5: Robot trajectory and obstacle displacements while employing the robot model in (4). Figures correspond to $m = 1, \ldots, 5$ starting from the first row.

	m	Robot model (3)				Robot model (4)			
		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	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 I: Different statistics for the three domains. CPU denotes the computation time in seconds for the optimization problem in (5). For $m > 1$, J^r, J^o represent the added cost across each slice.



(a) **(b)** **(c)**

Fig. 6: (a) A representative minimum obstacle displacement domain. (b)-(c) Increasing the obstacle displacement factor (J^o) weight from 10 to 1000, the humanoid finds a path with lower obstacle displacement but with larger path length.

VII. Conclusion

This paper looked into the MOD problem from a mobile robot navigation perspective. In addition to an exact method, a horizon slicing approach is presented which computes an $m-$approximate solution with high computational gain.

The evaluation domains used in this paper has been abstract. Yet, this is by no means a limitation and there exists many practical applications to the minimum obstacle displacement problem. Consider a humanoid navigating a cluttered table environment as seen in Fig. 6a. The humanoid is asked to move to the other room but its path is blocked by the tables. Employing the robot model (3) with $m = 2$ and $\alpha^{re} = 1$, $\alpha^r = 1$, $\alpha^o = 10$, a 7-obstacle solution with a displacement magnitude of 4.4 m is obtained (see Fig. 6b). Penalizing large displacements by increasing α^o from 10 to 1000, a longer path with a 2-obstacle solution (see Fig. 6c) is obtained with a lower displacement magnitude of 0.4 m. The cost c_k for the last time step is the deviation of the robot location from its goal (6). While splitting the objective function in (14) and (15), the deviation from the goal is naturally incorporated for in the last slice (sub-problem.) However, in each sub-problem the robot needs to keep track of its goal. Thus, for all the other sub-problems, we add the term $\alpha^r \alpha^{re} ||\mathbf{x}_{(i-1)T'} - \mathbf{x}^g||^2$ with $2 <= i <= m-1$. In the computation of ϵ and the values J^r, J^o for $m > 1$ in Table I, this term is ignored to provide an exact comparison to the optimization with $m = 1$.

In this work we do not deal with interacting obstacles, that is, obstacles are themselves not collision free. This can be achieved by adding the collision avoidance constraint among obstacles. However, for large obstacle number, this increases the computational burden due to addition of binary variables to achieve the linear collision avoidance constraints. Currently, we do not see how to move the obstacles and assume that the computed displacements can be achieved. For example, an obstacle might be too heavy for moving even though it gives a minimum displacement solution. This aspect may be incorporated with heavy obstacles being penalized more by giving a higher weight than the other obstacles. It is also interesting to include additional aspects such as work done by the robot which also absorbs the previous concern with force required to move the obstacles.

References

[1] K. Hauser, "The minimum constraint removal problem with three robotics applications," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 5–17, 2014.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.
[3] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.
[4] D. Nieuwenhuisen, A. F. van der Stappen, and M. H. Overmars, "An effective framework for path planning amidst movable obstacles," in *Algorithmic Foundation of Robotics VII*, pp. 87–102, Springer, 2008.
[5] J. Van Den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha, "Path planning among movable obstacles: a probabilistically complete approach," in *Workshop on the Algorithmic Foundations of Robotics VIII, WAFR, Guanajuato, Mexico*, pp. 599–614, Springer, 2009.
[6] G. Wilfong, "Motion planning in the presence of movable obstacles," *Annals of Mathematics and Artificial Intelligence*, vol. 3, no. 1, pp. 131–150, 1991.
[7] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proceedings 2007 IEEE international conference on robotics and automation*, pp. 3327–3332, IEEE, 2007.
[8] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," in *Robotics: Science and systems VII*, 2011.
[9] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement.," in *Robotics: Science and Systems*, vol. 1123, 2015.
[10] H. Karami, A. Thomas, and F. Mastrogiovanni, "Task Allocation for Multi-robot Task and Motion Planning: A Case for Object Picking in Cluttered Workspaces," in *AIxIA 2021 – Advances in Artificial Intelligence*, (Cham), pp. 3–17, Springer International Publishing, 2022.
[11] L. I. R. Castro, P. Chaudhari, J. Tůmová, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *52nd IEEE Conference on Decision and Control*, pp. 3217–3224, IEEE, 2013.
[12] B. Xu and H. Min, "Solving minimum constraint removal (mcr) problem using a social-force-model-based ant colony algorithm," *Applied Soft Computing*, vol. 43, pp. 553–560, 2016.
[13] A. Krontiris and K. E. Bekris, "Trade-off in the computation of minimum constraint removal paths for manipulation planning," *Advanced robotics*, vol. 31, no. 23-24, pp. 1313–1324, 2017.
[14] B. Xu, L. Chen, and K. Xu, "Deep learning algorithm for minimum constraint removal (mcr) problem," in *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*, pp. 52–56, 2020.
[15] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.
[16] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 639–646, IEEE, 2014.
[17] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach," in *Robotics: Science and Systems*, 2016.
[18] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "FFRob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.
[19] A. Thomas, F. Mastrogiovanni, and M. Baglietto, "MPTP: Motion-planning-aware task planning for navigation in belief space," *Robotics and Autonomous Systems*, vol. 141, p. 103786, 2021.
[20] L. Zhang, Y. J. Kim, and D. Manocha, "A simple path non-existence algorithm using c-obstacle query," in *Algorithmic Foundation of Robotics VII*, pp. 269–284, Springer, 2008.
[21] J. Basch, L. J. Guibas, D. Hsu, and A. T. Nguyen, "Disconnection proofs for motion planning," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 2, pp. 1765–1772, IEEE, 2001.
[22] S. Li and N. T. Dantam, "Learning proofs of motion planning infeasibility," in *Robotics: Science and Systems*, 2021.
[23] A. Thomas and F. Mastrogiovanni, "Minimum Displacement Motion Planning for Movable Obstacles," in *Intelligent Autonomous Systems 17*, (Cham), pp. 155–166, Springer Nature Switzerland, 2023.
[24] K. Hauser, "Minimum constraint displacement motion planning," in *Proceedings of Robotics: Science and Systems IX*, (Berlin, Germany), June 2013.
[25] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.
[26] V. V. Naik, *Mixed-integer quadratic programming algorithms for embedded control and estimation*. PhD thesis, IMT School for Advanced Studies Lucca, 2018.
[27] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in matlab," in *In Proceedings of the CACSD Conference*, (Taipei, Taiwan), 2004.