International Journal of Robotics Research (Special Issue on WAFR '06) 27(12):1296-1307
November, 2008.

Planning Among Movable Obstacles with Artificial Constraints

Mike Stilman and James J. Kuffner

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213 USA
[robot,jkuffner]@cmu.edu

Summary. This paper presents artificial constraints as a method for guiding heuristic search in the computationally challenging domain of motion planning among movable obstacles. The robot is permitted to manipulate unspecified obstacles in order to create space for a path. A plan is an ordered sequence of paths for robot motion and object manipulation. We show that under monotone assumptions, anticipating future manipulation paths results in constraints on both the choice of objects and their placements at earlier stages in the plan. We present an algorithm that uses this observation to incrementally reduce the search space and quickly find solutions to previously unsolved classes of movable obstacle problems. Our planner is developed for arbitrary robot geometry and kinematics. It is presented with an implementation for the domain of navigation among movable obstacles.

1 Introduction

A robot that can move obstacles out of its way is capable of more autonomous tasks. For example, in Figure 1, the robot cannot directly plan a path to the goal. By manipulating four objects, the robot changes its configuration space and opens free space for a path. This capacity comes at the cost of computational complexity. We explore a method for allowing robots to constrain their action space and create computationally manageable search spaces.

A simple path planning task in the movable obstacle domain becomes a complex manipulation planning problem with a partially specified goal. The robot can change its own configuration and the configurations of other objects. Each of these changes alters the workspace of the robot by increasing or decreasing the free space for future motions. The size of the search space is exponential in the number of movable objects. Furthermore, the branching factor of forward search is linear in the number of all possible world interactions [1]. A simplified variant of this domain involving only one movable obstacle is NP-hard.[2] More recent results demonstrated NP-hardness results for trivial problems where square blocks are pushed in block-size increments on a planar grid.[3]

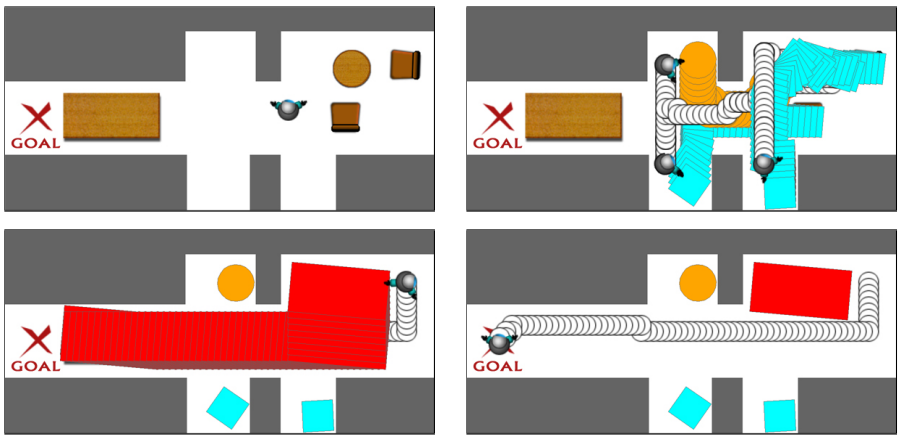


Fig. 1. A simulated solution to a problem of Navigation Among Movable Obstacles. The robot is instructed to reach the goal. After constructing a plan, it first moves the three smaller objects to the niches. The robot uses the new free space to move the table. Finally it clears a path and navigates to the goal.

In this paper we show that allowing one interaction with each object and reverse planning let the robot constrain its initial search space. We do not escape the curse of dimensionality. The proposed method of *artificial constraints* enables fast heuristic search in a domain where standard proximity heuristics provide little or no insight. We demonstrate that our method is directly applicable to robot tasks in a simulated domain. Furthermore, we introduce a problem formulation and runtime analysis that form a basis for future work.

2 Related Work

Obstacles moving along specified trajectories is a problem addressed by bounding the velocities of the obstacles and augmenting the configuration space with time.[4] A point in the free space ensures that a configuration is valid at the given time in which it takes place. This approach has been extended to kinodynamic domains, [5] as well as real-time deformable plans.[6] These algorithms do not allow the robot to affect the world.

Initial work in coordinating robot motions can be found in [7, 8, 9]. Most recent research that deals with robots repositioning *multiple objects* is in assembly planning. Assembly planners focus on separating a collection of parts and typically ignore the robot/manipulator. Domain operators also allow unassembled parts to be removed to "infinity."[10, 11]

In the *movable obstacle* domain, objects cannot move unless manipulated by the robot. The motion of the objects is constrained to the workspace of the robot, while the robot is constrained to move along collision-free paths. *Rearrangement planning* is the domain that is most closely related. [12, 13, 14] The final configurations of all objects are specified, and the robot must find coordinated transport paths. For instance, when a manipulation path

to the goal collides with other objects, [14] heuristically selects intermediate configurations for interfering obstacles.

In our domain, final configurations for objects are unspecified. Hence the robot must decide not only where to move objects but which objects must be moved. [15] searched a graph of robot paths, allowing objects to be pushed away from the robot trajectory. This method is effective on small problems, but easily encounters local minima. [1] and [16] propose to consider joining regions of robot free space and constructing graphs of interfering obstacles respectively. Neither planner handles objects that interfere with the motion of other objects. [17] ignores the robot, but offers some insight into graph-based chronological and spatial coordination of movable objects.

[15, 1, 16, 14] were developed for mobile robots. Our work addresses the problem generally for any kinematic structure of the robot. This is important when considering manipulation problems where robot geometry varies significantly for different portions of the workspace. We will base our domain on configuration space operators first described in [12]. Our constraint approach is related to [8], however we do not assume a priority on object motions, rather we must search the space of object choices and orders.

3 Movable Obstacle Domain

In this section, we develop a geometric model for movable obstacles. Our choice of space and operators make the presented approach general for any robot kinematics in the framework of rigid body motion and prehensile manipulation. Section 7 gives an example of how the tools developed in this framework can be applied to a specific robot problem.

Consider a path planning problem in a 2D or 3D Euclidian space that contains a set of fixed objects $\mathbf{O}_F = \{F_1, \ldots, F_f\}$ and a set of movable objects $\mathbf{O}_M = \{O_1, \ldots, O_m\}$. The space also contains an n degree of freedom robot, R. While paths are not explicitly parameterized by time, we will use the variable t to refer to a chronological ordering on states and operations. A world state at time-step t is the tuple consisting of t, the robot configuration r^t and the configuration q_i^t of each movable object: $W^t = (t, r^t, q_1^t, q_2^t, \ldots, q_m^t)$.

Let $\mathcal{C}_W$ be the space of all possible W^t. We permit the robot to move one object at a time. Consequently, we are interested in subspaces or *slices* of $\mathcal{C}_W$:

$\mathcal{C}_R(W^t) = (\{r\}, q_1^t, q_2^t, \ldots, q_m^t)$ - the slice of robot configurations, and
$\mathcal{C}_{O_i}(W^t) = (r^t, q_1^t, q_2^t, \ldots, \{q_i\}, \ldots, q_m^t)$ - configurations of object O_i.

Observe that any slice is parameterized by the positions of other objects. Following [18] we define free space to be the subspace of collision free configurations. First consider the configuration space obstacles (CO). Let $A(q) = \{x \in \mathbb{R}^k | x \text{ is a point of object } A \text{ in configuration } q\}$. For any set of points S in $\mathbb{R}^k$, a configuration space obstacle in $\mathcal{C}_B$ is the set: $CO_B(S) = \{p \in \mathcal{C}_B | B(p) \cap S \neq \emptyset\}$. Let q be a configuration of object A and p be a configuration of object B. Since no two objects can occupy the same space in $\mathbb{R}^n$, CO is symmetric:

$$p \in CO_B(A(q)) \Rightarrow B(p) \cap A(q) \neq \emptyset \Rightarrow q \in CO_A(B(p)) \tag{1}$$

To simplify notation, let $CO_R^{q_i} = CO_R(O_i(q_i))$ and $CO_{O_j}^{q_i} = CO_{O_j}(O_i(q_i))$ represent obstacles due to O_i in $\mathcal{C}_R$ and $\mathcal{C}_{O_j}$ respectively.

Let $\overline{CO_A(B)}$ be the complement of $CO_A(B)$ in $\mathcal{C}_A$. The free space of a movable object, $\mathcal{C}_A^{free}(W^t)$, is the set of configurations where the object is not in collision with fixed or movable obstacles.

$$\mathcal{C}_{O_i}^{free}(W^t) = \bigcap_k \overline{CO_{O_i}(F_k)} \bigcap_{O_j \neq O_i} \overline{CO_{O_i}^{q_j^t}} \tag{2}$$

Collisions between a moving object and the robot are treated separately from Eq. 2 since the motion of an object also implies the motion of the robot. $\mathcal{C}_R^{free}(W^t)$ is expressed analogously in terms of CO_R.

In spaces with external forces, such as gravity, objects will not remain static in arbitrary configurations. Manipulated objects must be released in *placement* configurations $\mathcal{C}_{O_i}^{place}(W^t) \subseteq \mathcal{C}_{O_i}^{free}(W^t)$. When solving three dimensional problems, we propose form closure to develop this set. In our two dimensional examples, we assume gravity is orthogonal to the object plane and hence $\mathcal{C}_{O_i}^{place}(W^t) = \mathcal{C}_{O_i}^{free}(W^t)$.

Having defined the sets of free configurations for the robot and movable objects, we now address the allowable interactions between the robot and the environment. Following [12], we define two parameterized operators on the $\mathcal{C}_{space}$: *Transit* and *Transfer*. *Transit* creates a path for the robot. *Transfer* represents the motion of the robot and a single movable object.

TRANSIT: We first define a continuous path τ in the configuration space of the robot: $\tau : [0,1] \rightarrow r$ for $r \in \mathcal{C}_R$. $\tau(r_i, r_j)$ will shorten the notation for a path where $\tau[0] = r_i$ and $\tau[1] = r_j$. The $Transit$ operator is a function that maps a world state and path to another world state.

$$Transit : (W^t, \tau(r^t, r^{t+1})) \rightarrow W^{t+1} \tag{3}$$

This operator is valid if and only if the following condition holds:

$$\tau(s) \in \mathcal{C}_R^{free}(W^t) \qquad \forall s \in [0,1] \tag{4}$$

TRANSFER: When an object is rigidly grasped, its configuration is fully determined by a transformation of the generalized pose of the robot end effector. $K : \mathcal{C}_R \rightarrow x$ ($x \in \mathbb{R}^n$) is the kinematic mapping of robot configurations to end effector positions/orientations. We will consider a discrete or sampled set of grasps for each movable object: $GS(O_i) = \{\mathbf{G}_{O_i}\}$. Each $\mathbf{G}_{O_i}$ is a rigid transform from the robot pose to a configuration of O_i. $\mathbf{G}_{O_i}(K(r)) = q$ states that the robot configuration r grasps O_i in configuration q.

For any grasp $\mathbf{G}_{O_i}^k \in GS(O_i)$, the $Transfer$ operator maps a world state and a path in $\mathcal{C}_R$ to a new state where the robot and an object are displaced:

$$Transfer : (W^t, O_i, \mathbf{G}_{O_i}^k, \tau(r^t, r^{t+1})) \rightarrow W^{t+1} \tag{5}$$

Notice that for any robot path τ we can compute τ_{O_i} for the object as the path $\tau_{O_i} = \mathbf{G}_{O_i}(K(\tau))$. A valid $Transfer$ operator must satisfy:

$$\tau(s) \in \mathcal{C}_R^{free}(W^t) \cup CO_R^{q_i^t} \quad \tau_{O_i}(s) \in \mathcal{C}_{O_i}^{free}(W^t) \qquad \forall s \in [0,1] \tag{6}$$

$$\tau_{O_i}(0) = q_i^t \tag{7}$$

$$\tau_{O_i}(1) \in \mathcal{C}_{O_i}^{place}(W^t) \tag{8}$$

$$R(\tau(s)) \cap O_i(\tau_{O_i}(s)) = \emptyset \qquad \forall s \in [0,1] \tag{9}$$

Eq. 8 requires that the final configuration of the object be statically stable. Eq. 9 ensures that the robot does not collide with obstacle O_i.

4 Motions of Multiple Objects

The problems we are interested in are realistic domains with numerous movable objects. Due to the dimension of these spaces, finding meaningful sub-domains and heuristics takes precedence over completeness. In earlier work [1] we observed that $\mathcal{C}_R^{free}$ can be partitioned into disjoint subsets $\{C_1, C_2, \ldots, C_d\}$ such that a robot in configuration $r_i \in C_i$ can access any configuration in C_i via a $Transit$ operation but no configuration in $C_j \neq C_i$.

Our planner detected objects that could be moved in order to give the robot access to other components of $\mathcal{C}_R^{free}$. For two subsets $C_i, C_j \in \mathcal{C}_R^{free}(W^t)$ and a border obstacle O_l we pursued a k-length sequence of $Transit$ and $Transfer$ operations that yield a merged component $C_{mrg} \subset \mathcal{C}_R^{free}(W^{t+k})$:

$$C_{mrg} = (C_i \cup C_j \bigcup CO_R^{q_l^t}) \bigcap \overline{CO_R^{q_l^{t+k}}} \tag{10}$$

$$\forall r_i, r_j \in C_{mrg} \text{ there exists } \tau(r_i, r_j) \text{ s.t. } \forall s(\tau[s] \in C_{mrg}) \tag{11}$$

Based on the concept of connecting free space components, we defined the class of *linear problems* (LP). A problem has a linear solution when there exists a sequence of free space components $\{C_1, C_2, \ldots, C_n\}$ such that merging adjacent components C_i and C_j does not constrain the C_{space} required to merge adjacent C_k and C_l where $(i < j \leq k < l)$. [1] presented a resolution complete algorithm for problems in L_1, where only one object must be displaced to merge two components.

Extending [1] to L_k problems where up to k objects may be moved to connect free space components is challenging even for $k = 2$. In the best case, every robot path between two components would pass through two objects, O_1 and O_2, allowing the planner to locally search the joint motion space of size $|O_1| \times |O_2|$. However, as seen in Figure 1, the path between C_i, $C_j \in \mathcal{C}_R^{free}$ might only pass through one object (the table). A complete L_2 planner must consider all possibilities for the choice of second object. In general, for L_k problems, we may need to enumerate 2^{k-1} possible sets of objects that do not directly interfere with a path to the goal.

4.1 Proposed Hierarchy

In order to manage the increased complexity when local search requires the motion of multiple objects, we propose further classification of the movable

obstacle domain to *monotone* plans. In assembly planning, monotone plans refer to plans where each application of an operator yields a subassembly that is part of the final assembly [11]. We do not enforce a final assembly and define *monotone plans* as those in which a transferred object cannot be moved again:

$$W^{t+1} = Transfer(W^t, O_i, \mathbf{G}^k_{O_i}, \tau) \quad \Rightarrow \quad q_i^T = \tau_{O_i}(1) \quad (T > t) \quad (12)$$

Monotone search decouples the joint motion space of objects into individual path plans. The search must decide which objects to displace, the $Transfer$ paths for each object and the ordering of object motion.

Notice that any plan can be expressed as a sequence of monotone plans:

$$Plan_{NM} = \ldots, \tau_1, (O_i, \tau_2), \tau_3, (O_j, \tau_4), \tau_5, (O_i, \tau_6), \tau_7, \ldots \equiv$$
$$Plan_{M_1} = \ldots, \tau_1, (O_i, \tau_2), \tau_3, (O_j, \tau_4), \tau_5 \text{ and } Plan_{M_2} = (O_i, \tau_6), \tau_7, \ldots (13)$$

Let W^6 be the world state after the operation $Transit(W^5, \tau_5)$, prior to the second displacement of O_i. We refer to W^6 as an intermediate world state. A problem can be characterized in its *non-monotone degree* by the number of intermediate states necessary to construct a sequence of monotone plans.

We propose the following classes of problems:

- L_k Linear problems where components of $\mathcal{C}_R^{free}$ can be connected independently. k is the maximum number of objects that must be displaced to connect two components.
- NL Non-linear problems that require the planner to consider interactions between keyholes.
- M Monotone problems where each object needs to only be displaced once throughout the plan.
- NM_i Non-monotone problems that can be expressed as i monotone problems with intermediate states.

A planner can operate in the space of one or two of these classes. For instance a planner in L_3NM_6 would seek linear solutions that require manipulating at most three objects and using six intermediate states to merge two free space components. Our proposed algorithm operates in L_kM.

5 Artificial Constraints

The monotone class of problems helps organize the study of movable objects. It still preserves a number of the computational challenges of our domain. The planner determines a subset $\{O_1, \ldots, O'_m\} = \mathbf{O}'_M \subset \mathbf{O}_M$ of movable objects to displace. It constructs a valid set of paths $\{\tau_{O_1}, \ldots, \tau_{O'_m}\}$ for displacing the objects and $\{\tau_1, \ldots, \tau_{m+1}\}$ $Transit$ operations between grasps. Additionally, the planner decides an ordering for object motion. This section will analyze the retained problem complexity and present our solution.

5.1 Complexity of Forward Search

Suppose we were to perform a standard forward search of obstacle motion. In the monotone case, we do not need to consider all possible $Transit$ and $Transfer$ paths. At each time-step t we select an object O_i for motion and a goal configuration $q_i^{t+2} \in \mathcal{C}_{O_i}^{place}(W^{t+2})$. We verify that there exists a robot configuration $r^{t+1} \in \mathcal{C}_R^{free}(W^t)$ that satisfies $q_i^t = \mathbf{G}_{O_i}^k(K(r^{t+1}))$ for some k. Additionally, we check the existence of valid paths:

$$\begin{aligned} & Transit(W^t, \tau_1(r^t, r^{t+1})) \text{ and} \\ & Transfer(W^{t+1}, O_i, \mathbf{G}_{O_i}^k, \tau_2(r^{t+1}, r^{t+2})) \\ \text{such that} \quad & q_i^{t+2} = \mathbf{G}_{O_i}^k(K(r^{t+2})) \end{aligned} \tag{14}$$

Assume that verification could be performed in constant time, and that the number of placements is $O(d^n)$, where d is the resolution of each of the n dimensions of $\mathcal{C}_{O_i}$. Typically, $n = 3$ or 6. At $t = 0$, this algorithm would select from m objects and d^n configurations for each object: $O(md^n)$. Expanding the search to depth 2, there are now $m-1$ objects and d^n placements for each object: $O(md^n \times (m-1)d^n)$. This algorithm has an asymptotic runtime of $O(md^n \times (m-1)d^n \times ... \times 2d^n \times d^n) = O(m!d^{nm})$.

The difficulty lies in finding an informed heuristic for the exponentially large space of object placements. A *good* placement for the object is one that respects the motion of subsequent obstacles. Since the motion of future objects is postponed in the search, good placements are unknown.

5.2 Reverse Search

Reverse planning is common in *assembly* problems. However, the implementation and motivation of reverse planning is different in our domain. Assembly planners have fixed goal configurations for all objects in which the motion of the objects is typically highly constrained. Consequently, the reverse search space has a much smaller branching factor due to *actual constraints*.

In the domain of movable obstacles, the final configuration is not predetermined, hence object motion must be planned from the initial state. Search reversal is performed in regard to the ordering of object motions (i.e. a $Transfer$ of the last object is performed as the first step of the search). At the start of search, the branching factor is large due to the non-existence of goal configurations. However, *artificial constraints* yield significant space reduction when searching prior motions.

Artificial Constraints

Let W^0 be the initial world state. Assume that at some future time step $t > 0$, the robot will perform a $Transit(W^t, \tau^t(r^t, r^{t+1}))$ operation. This operation is valid only when $\tau^t(s) \in \mathcal{C}_R^{free}(W^t)$ (Eq. 4). Let q_j^t be the configuration of obstacle O_j at time t. By the definition of free configuration space (Eq. 2):

$$\tau^t(s) \notin CO_R(O_j(q_j^t)) \tag{15}$$

Due to the symmetry of CO (Eq. 1), we can invert this relationship.

$$q_j^t \notin CO_{O_j}(R(\tau^t(s))) \qquad \forall s \in [0,1] \tag{16}$$

The robot motion along τ^t defines a *swept volume* in $\mathbb{R}^n$. Let $V(\tau^t)$ be the volume of points occupied by the robot during its traversal of τ^t:

$$Transit(W^t, \tau^t(r^t, r^{t+1})) \rightarrow V(\tau^t) = \bigcup_{s\in[0,1]} R(\tau^t(s)) \tag{17}$$

$$q_j^t \notin CO_{O_j}(V(\tau^t)) \tag{18}$$

Analogously, if we assume a valid $Transfer(W^t, O_i, \mathbf{G}_{O_i}^k, \tau(r^t, r^{t+1}))$ at step t ($t > 0$), we would define $V(\tau^t, O_i, \mathbf{G}_{O_i}^k)$ as the volume of points occupied by the robot and the object during their joint motion:

$$Transfer(W^t, O_i, \mathbf{G}_{O_i}^k, \tau^t(r^t, r^{t+1})) \rightarrow$$
$$V(\tau^t, O_i, \mathbf{G}_{O_i}^k) = \bigcup_{s\in[0,1]} [R(\tau^t(s)) \cup O_i(\tau_{O_i}^t(s))] \tag{19}$$

From Eq. 6 we find

$$\tau^t(s) \notin CO_R(O_j(q_j^t)) \qquad \tau_{O_i}^t(s) \notin CO_{O_i}(O_j(q_j^t))) \qquad (j \neq i). \tag{20}$$

Due to the symmetry of CO:

$$q_j^t \notin CO_{O_j}[R(\tau^t(s)) \cup O_i(\tau_{O_i}^t(s))] \qquad \forall s \in [0,1] \tag{21}$$

$$q_j^t \notin CO_{O_j}(V(\tau^t, O_i, \mathbf{G}_{O_i}^k)) \tag{22}$$

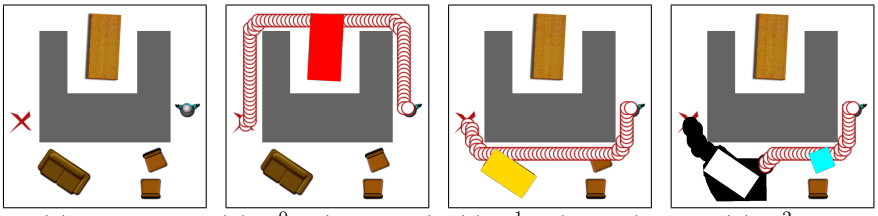
Eq. 18 and 22 indicate that the swept volume of any $Transit$ or $Transfer$ operation in W^t places a constraint on the configurations of movable objects: $V^t = V(\tau^t)$ or $V(\tau^t, O_i, \mathbf{G}_{O_i}^k)$ respectively. Since objects remain fixed unless moved by $Transfer$, then for some final time T:

$$q_j^0 \notin CO_{O_j}(V^T) \text{ or there exists a time } t(0 \leq t < T) \text{ such that}$$
$$Transfer(W^t, O_j, \mathbf{G}_{O_j}^k \tau^t(r^t, r^{t+1})) \text{ and } \tau_{O_j}^t(1) \notin CO_{O_j}(V^T) \tag{23}$$

Due to our assumption of monotone plans, if the initial configuration of an obstacle conflicts with V^T, there is exactly one $Transfer$ operator that displaces it to a non-conflicting configuration at some time-step t ($t < T$).

6 Algorithm

In order to apply the method of artificial constraints, our planner consists of two modules: *obstacle identification* and *constraint resolution*. Obstacle identification decides the last object that will be manipulated prior to reaching



(a) Problem (b) $\mathcal{P}^0_{last}$ (Unsolved) (c) $\mathcal{P}^1_{last}$ (Solved) (d) $\mathcal{P}^2_{last}$

Fig. 2. $\mathcal{P}_{last}$ selects the last object for manipulation by the robot. The planner is called for alternative selections (c), and for preceding subgoals (d).

the goal or a subgoal. Constraint resolution plans a $Transfer$ path for this object and the following $Transit$ to the goal. The two paths form artificial constraints. We detect objects that violate the constraints in W^0 and recursively plan corresponding $Transfer$ and $Transit$ operations. The first grasping configuration identified by a successful resolution step is used as the preceding subgoal for obstacle identification. Both modules backtrack on their choices when the algorithm fails to resolve the constraints.

6.1 Obstacle Identification

The search is initialized by a constrained relaxed planner $\mathcal{P}_{last}$. [1] $O_L \leftarrow \mathcal{P}_{last}$ operates in $\mathcal{C}_R$. It is permitted to pass through movable configuration space obstacles with a heuristic one-time cost for entering any object. $\mathcal{P}_{last}$ finds a path to the goal and selects the last colliding obstacle, O_L, to schedule for manipulation. In Figure 2(b), $\mathcal{P}^0_{last}$ selects the table.

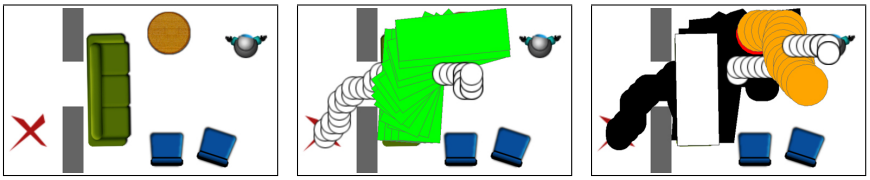
Constraint resolution, described in Section 6.2, validates the heuristic selection with a sequence of $Transit$ and $Transfer$ operations. If no such sequence is possible, $\mathcal{P}_{last}$ is called again, prohibiting any transition into $CO^{q^0_i}_R$. Since constraint resolution fails on the table, $\mathcal{P}^1_{last}$ selects the couch for motion in Figure 2(c). We ensure completeness over the selection of final objects by aggregating $\mathbf{O}_{avoid}$, a set of prohibited transitions for $\mathcal{P}_{last}$.[1]

When resolution is successful, $\mathcal{P}_{last}$ is called with the goal of reaching the initial grasping configuration identified by constraint resolution. Figure 2(d) shows that after successfully scheduling the manipulation of the couch, $\mathcal{P}^2_{last}$ selects the chair for motion. Finally, when $\mathcal{P}_{last}$ finds a collision-free path to the subgoal, the algorithm terminates successfully.

6.2 Constraint Resolution

Let T index the final time step of the plan and t be the current time step. We will maintain the following sets:

$\mathbf{O}^t_f$ - the set of objects O_i scheduled for manipulation after time t.
$\mathbf{O}^t_c$ - the set of objects scheduled for motion prior to time t.
$\mathbb{V}^t$ - the union of all artificial constraints $V^{t'}(t < t' < T)$.



(a) Problem (b) Manipulation Plan (c) Constraint Resolution

Fig. 3. (a) To free a goal path in W^T, $\mathcal{P}$ chooses to manipulate the couch in W^{T-2}. (b) The planner selects the manipulation of the couch that minimizes collision. (c) In W^{T-4} our planner manipulates the table to clear space for transferring the couch.

$\mathbb{V}^T$ is initialized as an empty volume of space. $\mathbf{O}_f^T$ and $\mathbf{O}_c^T$ are empty sets.

We begin by applying $O_L \leftarrow \mathcal{P}_{last}$ and adding O_L to $\mathbf{O}_c^T$. Constraint resolution attempts to move all objects from $\mathbf{O}_c$ to $\mathbf{O}_f$. Objects may be added to $\mathbf{O}_c$ when they interfere with manipulation. The following three procedures are performed recursively. Each iteration of recursion will plan from state W^t, such that operations that follow time step t are assumed to be known.

(1) Choosing an Obstacle and Grasp

First, we select an obstacle $O_d \in \mathbf{O}_c^t$. We then choose a grasp, $\mathbf{G}_{O_d}^k$ from a predefined set $\{\mathbf{G}^1, ..., \mathbf{G}^n\}$. Each grasp corresponds to a robot configuration r_{gi}. If the robot is redundant the space of inverse kinematic solutions is sampled, yielding a set of robot configurations $\{r_{g1}, r_{g2}, ..., r_{gn}\}$.

From the set of grasping configurations we select r^{t-2} such that for some k: $\mathbf{G}_{O_d}^k(r^{t-2}) = q_d^{t-2} = q_d^0$. The grasp transform specifies that the robot configuration r^{t-2} is grasping object O_d in the objects initial configuration.

We plan a path $\tau_g(r_0, r^{t-2})$ to verify that the grasp configuration can be reached by the robot without passing through previously scheduled obstacles in their initial configurations:

$$\tau_g(s) \notin (\bigcup_{O_i \in \mathbf{O}_f^t} CO_R^{q_i^0}) \cup CO_R^{q_d^0} \qquad \forall s \in [0,1] \tag{24}$$

If such a path does not exist, the planner searches over alternative grasps.

(2) Dual Planning for Transfer and Transit

The $Transit$ operation to the subsequent grasp, or goal, occurs after the $Transfer$ of an object. Chronologically it should be planned first. However, we have not yet determined the initial configuration for $Transit$ since it is equivalent to the final configuration of the $Transfer$ task. We propose assembling the $Transit$ path from two segments: τ_1 is a path from the initial grasp of the object to r^T and τ_2 is the $Transfer$ path of the object. The robot returns to its initial grasping configuration, r^{t-1}, during transit.

τ_1) Plan a partial path τ_1 from r^{t-2} to r^t. The path must not pass through any future scheduled obstacle:

$$\tau_1(s) \notin \bigcup_{O_i \in \mathbf{O}_f^t} CO_R^{q_i^0} \qquad \forall s \in [0,1] \tag{25}$$

Notice that taken alone this path is not intended for a $Transit$ operation. In the world state W^{t-2}, object O_d may still block this path. We choose this path heuristically to pass through the least number of objects in their initial configuration and minimize euclidian path length. If no such path is possible, a different grasp, r^{t-2}, is selected.

τ_2) Plan $Transfer(W^{t-2}, O_d, \mathbf{G}^k_{O_d}, \tau_2^{t-2})$. The robot configuration at the start of the plan is $\tau_2(0) = r^{t-2}$. The final configuration of the robot must be selected by the planner. Given that τ_2 maps to the object path $\mathbf{G}^k_{O_d}(\tau_2) \rightarrow \tau_{2O_d}$, we require the paths to adhere to the following constraints:

$$\tau_2(s) \notin \bigcup_{O_i \in \mathbf{O}_f^t} CO_R^{q_i^0} \quad \tau_{2O_d}(s) \notin \bigcup_{O_i \in \mathbf{O}_f^t} CO_{O_d}^{q_i^0} \quad \forall s \in [0,1] \tag{26}$$

$$\tau_{2O_d}(1) \notin CO_{O_d}[R(\tau_1(s)) \cup R(\tau_2(s))] \qquad \forall s \in [0,1] \tag{27}$$

$$\tau_{2O_d}(1) \notin CO_{O_d}(\mathbb{V}^t) \tag{28}$$

Eq. 26 states that the object and the robot may not pass through the configuration space obstacles of future scheduled objects. Eq. 27 states that the final configuration of O_d may not interfere with neither path segment τ_1 nor τ_2. Eq. 28 requires the final configuration of O_d to be consistent with the artificial constraints imposed by future motion.

Merging τ_1 and τ_2 into a single τ, we can define the operation $Transit(W^{t-1}, \tau)$. The transit is valid after the obstacle has been displaced.

(3) Composing Artificial Constraints

Having selected $Transfer$ and $Transit$ operations in W^t, we can advance the search to W^{t-2}. To do so, we will update the three sets described earlier:

$$\mathbf{O}_f^{t-2} \leftarrow \mathbf{O}_f^t \cup \{O_d\} \tag{29}$$

$$\begin{aligned} \mathbb{V}^{t-2} &\leftarrow \mathbb{V}^t \cup V(\tau_1) \cup V(\tau_2, O_d, \mathbf{G}^k_{O_d}) \\ &= \mathbb{V}^t \cup R(\tau_1(s)) \cup R(\tau_2(s)) \cup O_d(\tau_{2O_d}(s)) \quad \forall s \in [0,1] \end{aligned} \tag{30}$$

$$\mathbf{O}_c^{t-2} \leftarrow \{\, O_i \mid O_i \notin \mathbf{O}_f^{t-2} \wedge q_i^0 \in CO_{O_i}(\mathbb{V}^{t-2})\} \tag{31}$$

Eq. 29 fixes the configuration of O_d to the initial configuration and marks it as resolved in future states. Eq. 30 updates the artificial constraint to include the $Transfer$ and $Transit$ in W^{t-2} and W^{t-1} respectively. Eq. 31 updates the set of conflicting objects that must be moved earlier than W^{t-2} to resolve the constraints.

6.3 Depth First Search

Section 6.1 and 6.2 detailed the components of our planner. We now introduce pseudo-code that reflects the structure of the search. IDENTIFY-OBSTACLE is called to initialize the plan. The algorithm is implemented as depth first search to conserve space required for planning and help with the interpretability. □ indicates a successful base case while (NIL) reflects backtracking.

IDENTIFY-OBSTACLE$(r^t, (\mathbb{V}^t, \mathbf{O}_f, \mathbf{O}_c))$

$\mathbf{O}_{avoid} \leftarrow \emptyset$
while $O_L \leftarrow \mathcal{P}_{last}(W^0, r^t, \mathbf{O}_{avoid}) \neq$ NIL
do
  if $O_L = none$ **return** $\square$
  $\mathbf{O}_c \leftarrow \{O_L\}$
  $(Plan, r^{t-n}, (\mathbb{V}^{t-n}, \mathbf{O}_f^{t-n})) \leftarrow$ RESOLVE-CONSTRAINTS$(r^t, (\mathbb{V}^t, \mathbf{O}_f, \mathbf{O}_c))$
  if $Plan \neq$ NIL
    then $PastPlan \leftarrow$ IDENTIFY-OBSTACLE$(r^{t-n}, (\mathbb{V}^{t-n}, \mathbf{O}_f^{t-n}, \mathbf{O}_c^{t-n}))$
      if $PastPlan \neq$ NIL
        then return $(PastPlan$ **append** $Plan)$
  $\mathbf{O}_{avoid} \leftarrow \mathbf{O}_{avoid} \cup \{O_L\}$
return NIL

RESOLVE-CONSTRAINTS$(r^t, \mathbb{V}^t, \mathbf{O}_f^t, \mathbf{O}_c^t)$

if $\mathbf{O}_c^t = \emptyset$ **return** $\square$
for each $O_d \in \mathbf{O}_c$
do
  Choose $r^{t-2} : \mathbf{G}_{O_d}^k(r^{t-2}) = q_d^0$
    s.t. exists $\tau_g(r_0, r^{t-2})$ **satisfying** $Eq.\ 24$
  Choose $\tau_1(r^{t-2}, r^t)$
    Satisfying $Eq.\ 25$
  Choose $\tau_2(r^{t-2}, r^{t-1})$
    Satisfying $Eq.\ 26-28$
  if no valid choices
    then return NIL
  determine $(\mathbf{O}_f^{t-2}, \mathbb{V}^{t-2}, \mathbf{O}_c^{t-2})$ **by** $Eq.\ 29-31$
  $(Plan, r^{t-n}, (\mathbb{V}^{t-n}, \mathbf{O}_f^{t-n}, \mathbf{O}_c^{t-n})) \leftarrow$
      RESOLVE-CONSTRAINTS$(r^{t-2}, (\mathbb{V}^{t-2}, \mathbf{O}_f^{t-2}, \mathbf{O}_c^{t-2}))$
  if $Plan \neq$ NIL
    then $Plan$ **append** $Transfer(W^{t-2}, O_d, \mathbf{G}_{O_d}^k \tau_2(r^{t-2}, r^{t-1}))$
      $Plan$ **append** $Transit(W^{t-1}, \tau_2 + \tau_1)$
      return $(Plan, r^{t-n}, (\mathbb{V}^{t-n}, \mathbf{O}_f^{t-n}, \mathbf{O}_c^{t-n}))$
return NIL

7 Implementation

The algorithm described in this paper is entirely general for two and three dimensional spaces with arbitrary configuration spaces for the manipulator. In this section we will discuss our implementation of the algorithm in the domain of *Navigation Among Movable Obstacles* (NAMO) [1].

NAMO is two dimensional domain where obstacles are represented by polygons. The robot is a circular disc that can grasp objects when the center of the disc is at a given distance from pre-defined contact points. The domain is selected due to its interpretability and the simple property of object placement: $\mathcal{C}_{O_i}^{place}(W^t) \subset \mathcal{C}_{O_i}(W^t)$. The figures in this paper are constructed by the implemented planner in our NAMO simulation environment.

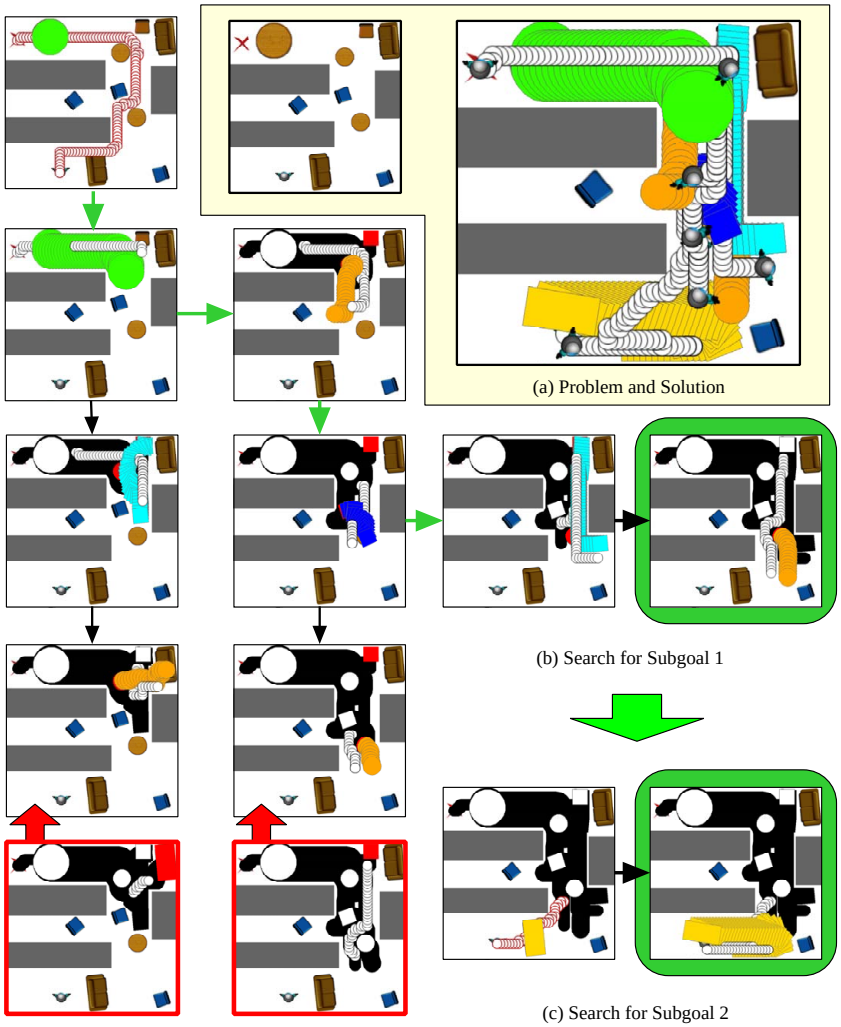


Fig. 4. A search tree for the given example. Large upward arrows indicate backtracking when an object cannot be resolved.

7.1 Planning Details

When constructing a plan for the NAMO domain, we directly apply the algorithm in Section 6 by selecting a computational representation of paths and artificial constraints:

- For paths, we choose a grid planner based on an evenly spaced discretization of $\mathcal{C}_R$. The robot configuration space has three dimensions: $(\mathbb{R} \times \mathbb{R} \times SO(2))$. Robot paths are planned in a matrix of resolution $(10cm, 10cm, 10^o)$ in each dimension respectively. This yields a simple, resolution complete search space.

- In the two dimensional domain, artificial constraints are sets of points in $\mathbb{R}^2$. Due to the rotation of objects, these sets could have complex curved boundaries. To reduce constraint verification (collision detection) to polygon intersection, we construct swept volumes using a local convex hull approximation method similar to [19] and [20]. We create local bounding polygons for the object and robot throughout the path.

- All obstacles and artificial constraints are rasterized in the form of an occupancy grid of the environment. Set membership in world coordinates is confirmed by verifying the occupancy of grid cells.

Choosing a $Transit$ path (τ_1) in $\mathcal{C}_R$ is performed using A^*. The heuristic is euclidian distance with a penalty for entering $CO_R(O_i)$ for the first instance of O_i along the path. This heuristic is selected to minimize the number of objects that will violate the artificial constraint in the preceding plan.

Analogously, since $Transfer$ paths (τ_2) have no explicit goal, we use best first search to make a selection. The first path and resulting state encountered by the search that satisfy the artificial constraints are chosen by the planner. Heuristically, we penalize states where robot or the transferred object enter movable configuration space obstacles.

7.2 Results

The implemented planner was tested on a number of examples, including all the figures presented in this paper. Table 1 summarizes the running times on an Intel Pentium M 1.6Ghz processor.

Table 1. Quantities of objects and running times for examples in Figures 1-4.

	Fig.1	Fig.2	Fig.3	Fig.4
# Objects	4	4	4	9
# Transferred	4	2	2	6
Planning Time	$0.77s$	$0.05s$	$0.10s$	2.08s

Of the presented examples, Fig. 1, 2 and 4 cannot be solved by existing planners [15, 1, 16]. In Fig. 3, the proposed method is asymptotically faster than [1] due to the early selection of $Transit$ paths as constraints in contrast to path validation during $Transfer$ search. However, this choice precludes completeness in the proposed implementation. In L_1 problems, [1] will discover remote $Transit$ paths that are not considered by the proposed implementation.

We find these results encouraging towards the implementation of this planner on a real robot system. Since the planner searches locally in the configuration space of the robot, the same algorithm can be applied directly to very high dimensional configuration spaces by replacing grid search methods with sampling-based alternatives.

7.3 Complexity

Since IDENTIFY-OBSTACLE never considers an obstacle more than once at any level of the search tree, it can generate at most $m!$ sequences. Each sequence

can contain m objects to be resolved by RESOLVE-CONSTRAINTS. A breadth first search of $\mathcal{C}_R$ of resolution d in n dimensions has runtime $O(d^n)$. The overall algorithm is asymptotically $O(m!d^n)$. This is a vast overestimate. In most cases only a few sequences will satisfy the conditions of the planner.

Notice, however, that each of three "Choose" statements in RESOLVE-CONSTRAINTS is an opportunity for backtracking (Lines 4, 6 and 8). Selecting a different simple path for $Transfer$ or $Transit$ will yield distinct artificial constraints for the remainder of the search. While enumerating all possible simple paths for robot motion and manipulation is computationally expensive, selecting a subset of these paths may prove to be valuable.

8 Conclusion and Future Work

In this paper, we have presented a general planner for movable obstacles in arbitrary configuration spaces. The heuristic methods of artificial constraints have proven to be fast and effective in resolving complex examples from the sample domain of Navigation Among Movable Obstacles.

Future work will consider the possibility of reducing the number of object orderings and examining alternative object paths. Some likely classes of heuristics are the following:

Accessibility Constraints - Currently we search through all orderings of objects that violate an artificial constraint. However, clearly some objects cannot be reached by the robot before others are moved. These objects must be moved at a later time-step.

Path Heuristics - Reverse search carries significant advantages to forward search in selecting alternative paths. Simply by finding paths that explore distinct, or distant, portions of space we would change the topology of artificial constraints and therefore open distinct possibilities for prior object placements.

In addition to the investigation of heuristics, it will be interesting to study the potential for using artificial constraints to determine the necessity of intermediate states. Doing so will enable planners to address the greater challenges of non-monotone problems.

9 Acknowledgements

We are grateful to C. G. Atkeson for his input and support in the development of this project. We thank the anonymous reviewers for their thorough reading and insightful comments. This research was partially supported by NSF grants DGE-0333420, ECS-0325383, ECS-0326095, and ANI-0224419.

References

1. M. Stilman and J.J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *IEEE/RAS Int. Conf. on Humanoid Robotics http://www.golems.org/NAMO*, pages 322–341. IEEE, Nov 2004.
2. Gordon Wilfong. Motion panning in the presence of movable obstacles. In *ACM Symp. Computat. Geometry*, pages 279–288. ACM Press, New York, NY, 1988.

3. E. Demaine, M. Demaine, and J. O'Rourke. Pushpush and push-1 are np-hard in 2d. In *12th Canadian Conference on Computational Geometry*, pages 211–219, Fredericton, New Brunswick, Canada, Aug 16-19 2000.
4. J. H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proc. 26th Annual Symposium on Foundations of Computer Science*, pages 144–154, Portland, Oregon, October 1985. IEEE.
5. D. Hsu, R. Kindel, J. C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The Int. J. of Robotics Research*, 21(3):233–255, March 2002. Sage Publications, Inc. Thousand Oaks, CA, USA.
6. Oliver Brock and Oussama Khatib. Elastic strips: Real-time path modification for mobile manipulation. In *International Symposium of Robotics Research*, pages 117–122, Hayama, Japan, October 1997. Springer Verlag.
7. S. Fortune, G. Wilfong, and Chee Yap. Coordinated motion of two robot arms. In *IEEE Int. Conf. on Robotics and Automation*, pages 1216 – 1223, Apr 1986.
8. Michael Erdmann and Tomas Lozano-Perez. On multiple moving objects. In *IEEE Int. Conf. on Robotics and Automation*, pages 1419–1424, Apr. 7-10 1986.
9. J. Schwartz and M. Sharir. On the piano movers' problem. iii: Coordinating the motion of several independent bodies. the special case of circular bodies moving amidst polygonal barriers. *Int. J. Robotics Research*, 2(3):46–74, 1983.
10. M. H. Goldwasser and R. Motwani. Complexity measures for assembly sequences. *The International Journal of Computational Geometry and Applications*, 9(4 & 5):371–418, April 1999. World Scientific Publishing Company.
11. Randall H. Wilson. *On Geometric Assembly Planning*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, USA, 1992. UMI Order Number: UMI Order No. GAX92-21686.
12. R. Alami, J.P. Laumond, and T. Sim'eon. Two manipulation planning algorithms. In K. Goldberg, D. Halperin, J. Latombe, and R. Wilson, editors, *Workshop on the Algorithmic Foundations of Robotics*, pages 109 – 125, San Francisco, California, United States, 1994. A. K. Peters, Ltd. Natick, MA, USA.
13. O. Ben-Shahar and E. Rivlin. Practical pushing planning for rearrangement tasks. *IEEE Trans. on Robotics and Automation*, 14(4):549–565, August 1998. IEEE Robotics & Automation Society.
14. J. Ota. Rearrangement of multiple movable objects. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, volume 2, pages 1962–1967, New Orleans, LA, April 2004. IEEE.
15. P.C.Chen and Y.K.Hwang. Pracitcal path planning among movable obstacles. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 444–449, Sacramento, CA, April 1991. IEEE.
16. K. Okada, A. Haneda, H. Nakai, M. Inaba, and H. Inoue. Environment manipulation planner for humaonid robots using task graph that generates action sequence. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, pages 1174–1179, Sendai, Japan, September 2004. IEEE.
17. T. Chadzelek, J. Eckstein, and E. Schömer. Heuristic motion planning with movable obstacles. In *8th Canadian Conference on Computational Geometry*, pages 131–136, Ottawa, Canada, Aug. 12-15 1996. Carleton University Press.
18. Tomas Lozano-Perez. Spatial planning: a configuration space approach. *IEEE Trans. Comput.*, 32(2):108–120, 1983. IEEE Computer society.
19. M. A Ganter. *Dynamic Collision Det. using Kinematics and Solid Modeling Techniques(Mechanical Engineering)*. PhD thesis, University of Wisconsin, 1985.
20. A. Foisy and V. Hayward. A safe swept-volume approach to collision detection. In *Robotics Research, Sixth International Symposium*, 1994.