The Feasible Transition Graph: Encoding Topology and Manipulation Constraints for Multirobot Push-Planning

Laura Lindzey[1], Ross A. Knepper[2], Howie Choset[1], and Siddhartha S. Srinivasa[1]

[1] Carnegie Mellon University, Pittsburgh PA 15213, USA
[2] Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Abstract. Planning for multirobot manipulation in dense clutter becomes particularly challenging as the motion of the manipulated object causes the connectivity of the robots' free space to change. This paper introduces a data structure, the Feasible Transition Graph (FTG), and algorithms that solve such complex motion planning problems. We define an equivalence relation over object configurations based on the robots' free space connectivity. Within an equivalence class, the homogeneous multirobot motion planning problem is straightforward, which allows us to decouple the problems of composing feasible object motions and planning paths for individual robots. The FTG captures transitions among the equivalence classes and encodes constraints that must be satisfied for the robots to manipulate the object. From this data structure, we readily derive a complete planner to coordinate such motion. Finally, we show how to construct the FTG in some sample environments and discuss future adaptations to general environments.

1 Introduction

Much research has focused on manipulating objects using groups of small general-purpose robots rather than the traditional large single purpose machines in a context such as manufacturing [1, 3, 6, 15, 19]. There are a number of scenarios in factories and warehouses for which using a team of robots can save time or even perform a task that was impossible with a single robot. Examples include maneuvering a cargo pallet in an area packed with boxes and performing precision assembly of large products like airplanes.

In the first example, in a cluttered environment, transporting a bulky object can be quite awkward. Consider a forklift trying to make a 90° turn among tight corridors. It may be more efficient to set down the load, reposition, and then drive in the new direction. However, this repositioning could be literally infeasible due to the load blocking a lone forklift's free space, or logistically infeasible given the time required to drive around other obstacles. In these cases, an additional forklift must already be in position to efficiently pick up and transport the load.

In the second example, we observe that current strategies for manufacturing large objects require factory fixtures called jigs. The goal is to have teams of robots replace the jigs and carry large parts in and around the assembly area, bringing them into contact when the assembly operation calls for it. This would allow the manufacturing plant to be brought up to be operational more quickly as well as be more flexible for reusing the infrastructure for other tasks. However, this will require robots that are able to maneuver large objects in a busy, cluttered factory floor.

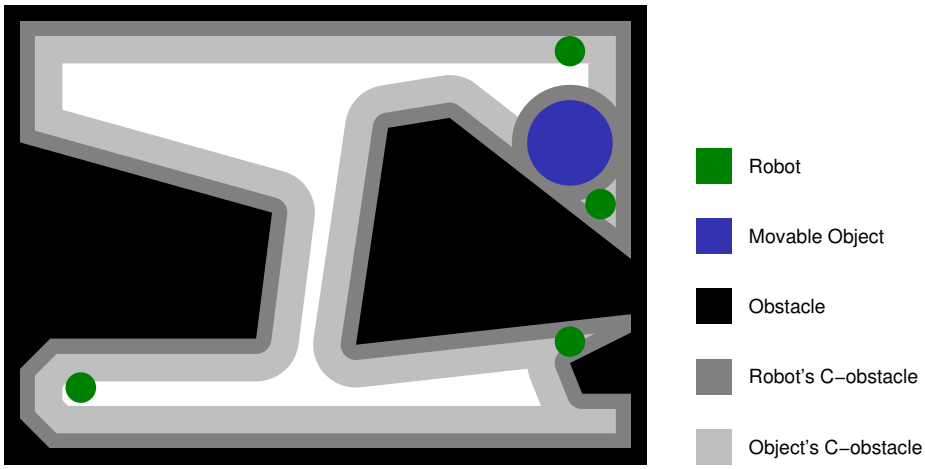


Fig. 1: Example environment for object pushing. The same color scheme is used throughout the paper.

One key challenge of these scenarios is that they require reasoning about how the manipulated object's location affects the manipulating robots' ability to move freely. They share that requirement with the *multirobot object manipulation* domain, which considers a class of problems where a group of mobile robots must work together to move an object from a start to a goal configuration (detailed in §3).

Consider cluttered environments such as in Fig. 1, with several robots pushing a large object in a maze-like environment. The motions of robots and the object are coupled both by obstruction and by rules for manipulating the object. In this paper, we explore the following questions:

P1 – Existence: Given start and goal configurations for the object, does a feasible plan exist for the robots to move the object?
P2 – Synthesis: Find a plan to move the object from a start to a goal location for given initial robot locations.
P3 – Optimization: What is the minimum number of robots required to push the object between two specific positions? What is the path-length-optimal feasible object path between two specific positions?
P4 – Minimalism: What is the minimum number of robots required to perform any feasible object path in the environment?

We introduce a novel representation, the *feasible transition graph*, and algorithms operating upon it that allow us to answer the above questions. We then discuss an implementation that solves these problems for a few simple types of environment and manipulation models.

We achieve these results by reformulating object pushing as a constrained minimization problem with constraints derived from two properties of the environment (§4). First, we require that robots obey the semantics of pushing, which we term *manipulation constraints*. These constraints determine how robots are able to maneuver the object. Next, as the object's motion changes the connectivity of the robots' free space,

we require that each robot must move deliberately among merging and splitting connected components. We call this *conservation of robots*. These two properties induce constraints on the number of robots occupying each connected component.

We organize these constraints in a graph-like structure called the feasible transition graph (§7), which makes it possible to solve multirobot planning problems (**P1**, **P2**, **P3**) with a graph search (§5). General minimum sufficient robots problems (**P4**) require an optimization over this graph (§6).

In the multirobot object pushing domain, even simple scenarios reveal much complexity. The feasible transition graph provides an abstraction that simplifies this complexity. A key future challenge is to tractably construct such graphs for complex, higher-dimensional problems (§8).

2 Related Work

Approaches to manipulation planning often consider a set of alternating transit and transfer actions. This makes it difficult to apply typical motion planning algorithms. Under some conditions, the manipulation planning problem can be split into two steps: first choosing a path for the object, and then finding robot paths that cause the object to follow that path [25]. This decomposition is similar to the division of multi-arm manipulation problems into *transit* and *transfer* tasks by Koga and Latombe [9]. Our work focuses exclusively on what is required to find a feasible object path, instead of on solving the navigation problem for individual robots. Once a path for the object has been found, it imposes a set of constraints on individual robot positions, from which motions can be easily computed. Robot paths that obey these constraints can be found using existing multirobot planning algorithms [5, 24].

Our multirobot manipulation task in clutter is closely related to navigation among movable obstacles. They both require reasoning about manipulating an object whose motion changes the connectivity of the robot's free configuration space [21, 25]. In this work, we use the observation that for determining whether a given manipulation action is feasible it is sufficient to explicitly track which portions of the robot's configuration space are occupied.

Significant previous work has focused on the mechanics of object pushing and the problem of how a team of robots can cause an object to follow a predetermined path. Lynch and Mason investigated the controllability of point- and line-contact pushing [11], [12]. More recently, de Berg and Gerrits [2] investigated how to compute paths for a team of robots to push an object along a given path among obstacles. Early work on the robot cooperation facet of this problem includes a demonstration that two robots are able to complete an obstacle free block pushing task more efficiently than a single robot [13] and an examination of how a group of robots can reorient blocks [16].

Caging is another method for solving the multirobot object pushing problem. Rather than alternating transit and transfer actions, robot actions are chosen such that they approach the goal while obeying constraints guaranteeing that the object remain caged. This approach has resulted in complete algorithms for obstacle free environments [22], and moderately cluttered environments [4, 14, 20]. However, we consider environments with narrow passages where it is not physically possible to cage an object.

$\mathbf{O}$	$= \{O_i\}$, obstacles	P_{R_i}, $P_{\mathbf{R}}$	path of R_i, set of robot paths
$\mathbf{R}$	$= \{R_i\}$, robots	P_M, $\mathbf{P}_M$	path for M, set of all such paths
M	manipulated object	F_M, $\mathbf{F}_M$	feasible path for M, set of all such paths
EC	Equivalence Class	$Q_R^{free}(q_M)$	free configuration space of robot R with M at q_M
EG	Equivalence Graph	Q_M^{free}	free configuration space of M
FTG	Feasible Transition Graph	$N(Q)$	number of connected components in space Q
$C(n_i)$	constraints on $n_i \in FTG$	$N(q_M)$	shorthand for $N\left(Q_R^{free}(q_M)\right)$ for $q_M \in Q_M^{free}$
$m_{\alpha i}$	number of robots in α_i	$A(n_i)$	Possible assignments of robots for $n_i \in EG$
		α_i	i^{th} connected component of $Q_R^{free}(q_M)$ for $q_M \in \alpha$

3 Definitions and Problem Statement

Assume the workspace is a closed, bounded subset of $\mathbb{R}^2$, populated by obstacles $\mathbf{O} = \{O_i\}$. Identical robots $\mathbf{R} = \{R_i\}$ cooperate to manipulate the movable object M, and are able to perform two types of actions within this environment: *transit actions*, where they move within a connected component of their free configuration space; and *transfer actions*, where they maneuver M. A solution for the object manipulation problem consists of a path for M from a start configuration $q_{M,init}$ to a goal configuration $q_{M,goal}$ and a set of robot trajectories $P_{\mathbf{R}} = \{P_{R_1}, P_{R_2}, \dots\}$ that cooperate to move M along this path.

We consider the case of homogeneous robots, and define $Q_R^{free}(q_M)$ to be the free configuration space formed by any robot R_i moving among $\mathbf{O}$ with M at position q_M. Q_M^{free} is the free configuration space formed by M moving among obstacles $\mathbf{O}$. Let the continuous function $P_M : [0,1] \rightarrow Q_M^{free}$ be a path for the object, and the set of all such paths be $\mathbf{P}_M$.

In order to tractably reason about all possible object paths, we define an equivalence relation on object positions q_M such that any path can be broken down into a series of actions transitioning among equivalence classes (ECs). We say that two object configurations $q_{M,i}$ and $q_{M,j}$ are equivalent if there exists a continuous path $p \in \mathbf{P}_M$ parametrized by $s \in [0,1]$, with $p(0) = q_{M,i}$ and $p(1) = q_{M,j}$, along which $N(p(s))$ is held constant. Each EC α is associated with a set of connected components $\{\alpha_1, \alpha_2, \dots\}$, as shown in Fig. 2. We use $m_{\alpha i}$ to represent the number of robots occupying the connected component α_i, and define a function $N(Q)$ that returns the number of connected components in a configuration space Q. $N(q)$ is used as shorthand for $N(Q_R^{free}(q))$.

We define *feasible paths* to be the subset of object paths that the robots are able push the object along:

$$\mathbf{F}_M = \{p \in \mathbf{P}_M \mid \exists P_{\mathbf{R}} \text{ causing } M \text{ to follow } p.\}$$

For a path to be feasible, there must be sufficient space adjacent to the object for a robot throughout the course of the manipulation. Considering the environment in Fig. 2, no transition from $\xi \rightarrow \alpha$ is feasible because there is no space for a robot to the left of M. Similarly, any path from $\alpha \rightarrow \beta \rightarrow \alpha \rightarrow \xi$ is infeasible, despite each individual transition being feasible. This is because earlier transitions requires $m_{\alpha_1} \geq 1$, but the final transition requires $m_{\alpha_1} = 0$.

Robots can move freely within Q_R^{free}, so chaining together feasible block paths only requires keeping track of how many robots occupy each connected component of $Q_R^{free}(q_M)$, rather than the full cross product of $|\mathbf{R}|$ such spaces. Individual robot trajectories can then be derived from the block path. This relies on two assumptions. First, the robots are interchangeable, such that we do not have to consider how robot positions affect the connectivity for other robots. (The robots in a conflict would simply have their goal assignments swapped.) Second, we assume that robot packing density is not a limiting factor, which depends on the details of the world model.

4 Approach

In cluttered spaces, constraints on manipulation and robot location interact in complex ways. In this section, we present our approach to simplifying the analysis of pushing interactions in order to solve Problems **P1**–**P4**. The first data structure, the *Equivalence Graph* (EG), exposes the topological structure of the ECs as a function of movable object position. The second data structure, the *Feasible Transition Graph* (FTG), computes feasible kinematic motions, using the ECs from the EG for bookkeeping about robot occupancy. Implementation details for sample environments are presented in §7.

4.1 Constraints

The data structures proposed here require us to associate constraints with each transition of the object across an EC boundary. Recall that we have two types of constraints: manipulation and conservation of robots, as described in §1. These constraints prescribe the number of robots assigned to each connected component. For example, in the environment shown in Fig. 2, a transition from EC δ to β imposes the following constraints:

$$\begin{aligned} m_{\delta 1} &= m_{\beta 3} && \text{(conservation of robots)} \\ m_{\delta 2} &= m_{\beta 1} + m_{\beta 2} && \text{(conservation of robots)} \\ m_{\delta 1} &\geq 1 && \text{(push manipulation)} \\ m_{\beta 3} &\geq 1 && \text{(push manipulation).} \end{aligned}$$

Conservation of robots constraints deal with the splitting and merging of connected components of $Q_R^{free}(q_M)$ over time. These constraints, which are a function of the geometry of the environment alone, take four different forms which we describe with references to the ECs shown in Fig. 2. In the case of merging components, such as $\delta \rightarrow \varepsilon$, we have $m_{\delta 1} + m_{\delta 2} = m_{\varepsilon 1}$. For splitting components, such as $\alpha \rightarrow \beta$, we have $m_{\alpha 2} = m_{\beta 2} + m_{\beta 3}$. For the same transition, we have $m_{\alpha 1} = m_{\beta 1}$, for components involved in neither splitting nor merging. If a component has no associated robots in the next EC, such as for $\gamma \rightarrow \eta$, we have $m_{\gamma 2} = 0$.

Manipulation constraints require that every connected component, or set thereof, responsible for generating a transition is occupied. We say that a connected component is responsible for a transition if a robot occupying it would be able to push the object in the required direction. This leads to constraints in the form of $m_{\alpha i} \geq 1$. In the case

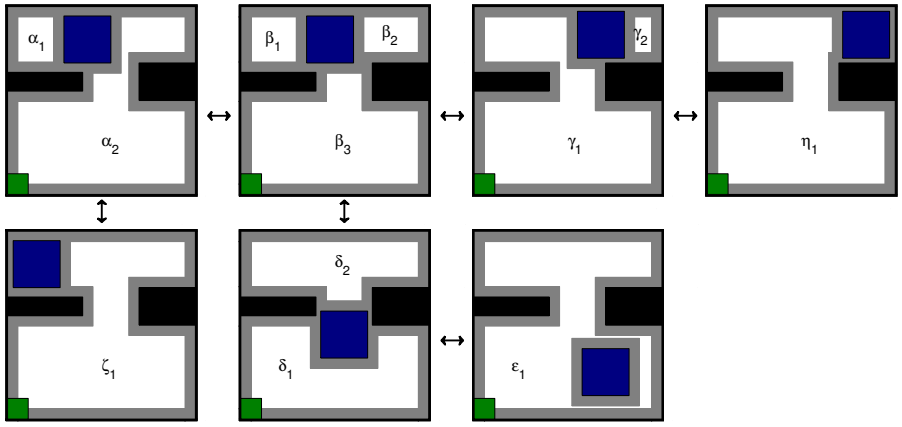


Fig. 2: An EG where each node is represented by an example object configuration. Arrows indicate neighboring ECs.

of multiple connected components able to execute the push, we only require that one of them be occupied, and the constraint takes the form $(m_{\alpha i} \geq 1) \vee (m_{\alpha j} \geq 1)$.

Changing the manipulation model requires changing how these manipulation constraints are defined and updating the feasibility checking to accommodate the different robot positions during motion. For example, if we wanted to require one robot pushing and one robot pulling, we would require that the connected components on either side of the object are occupied by at least one robot and that there is space at the start and end of the motion for both robots. Other possible configurations include allowing one robot to both push and pull, or requiring two robots pushing side-by-side to manipulate the object.

4.2 Equivalence Graph

The *equivalence graph* (EG) encodes a compact representation of the topology of the environment and the motion of the object. It is an undirected graph used to represent how the object's motion between ECs affects the connectivity of Q_R^{free}. Each node of the EG corresponds to an equivalence class (EC), as defined in §3. Every EC denotes a number of connected components of the robots' free configuration space, given alphanumeric labels in Fig. 2. The edges represent object motions that cause a transition between ECs, and are labeled with the corresponding conservation of robots constraints. Using the EG and an exact mapping $q_b \mapsto EC$, it is possible to determine the conservation of robots constraints involved for any object path P_M.

4.3 Feasible Transition Graph

The *feasible transition graph* (FTG) describes feasible object motions in the environment. It is a directed graph, reflecting the fact that transfer actions are not reversible in time. The nodes are object configurations, and edges are labeled with the constraints on connected component occupancy required for the associated object motion to be feasible. It has two key properties: any feasible object motion must map to a walk on the

FTG, and for any walk on the FTG, we must be able to determine which EC transitions have been crossed. If it is possible to exactly describe all such transitions, the resulting planner will be complete, and bounds on the number of robots required will be exact. Otherwise, it is possible to use a sampling-based approach to construct the FTG and obtain a probabilistically complete planner. When constructed, the FTG's nodes have no associated constraints; the Planning (§5) and Minimum Sufficient Robot (§6) algorithms both add annotations to the nodes of the FTG and propagate them through the FTG. These node annotations may represent constraints on robot assignments, denoted $C(n_i)$, or feasible robot assignments, denoted $A(n_i)$.

5 Planning

Given an initial object position M and robot positions $\mathbf{R}$, we wish to find a sequence of object pushes that cause the object to reach the goal location (**P2**). We present a roadmap-like planner that solves this problem. A roadmap planner uses a directed graph, where the nodes are configurations and the edges represent feasible paths between the configurations. It also requires that the graph be accessible/departible from any configuration and that it preserves connectivity [10]. We use the FTG as described in §4.3 as the roadmap, and give details for connecting start and goal positions in §7.3.

Nodes in the FTG may be labeled with an assignment of robots to connected components and/or a set of constraints. For **P2**, a labeling of robot assignments indicates that there is a feasible object path that could result in the robots moving from their given initial conditions to the indicated locations. A labeling of constraints indicates that if those constraints are met at that node, then there is a feasible block path from that node to the goal. A solution has been found when there exists a node with an assignment of robots that satisfies its constraints.

Possible robot assignments to connected components of $Q_R^{free}(q_M)$ propagate along the edges, starting with the provided initial conditions, and only change along edges that cross an EC boundary. The child node is assigned the set of all possible robot assignments to $\{m_{\alpha i}\}$ that satisfy the constraints on the transition and could result from starting with (one of) the parent node's robot assignment(s) and repartitioning the robots into connected components, if applicable. For example, consider the planning problem shown in Fig. 7b. The initial conditions are $A(n_{11}) = \{\varepsilon_1 = 3\}$. After a transition from $\varepsilon \to \delta$, we have $A(n_{12}) = \{(\delta_1, \delta_2) = (1,2), (2,2), (3,0)\}$. The partition $(\delta_1, \delta_2) = (0,3)$ was eliminated because it does not satisfy the constraints for the edge. In the case of a node with two parents, we add both sets of possible assignments to the set. This has a branching factor proportional to $\binom{|\mathbf{R}|}{|\alpha_i|}$. In practice, this may be reduced dramatically by the requirement that propagated assignments satisfy the edge constraints, and is bounded by the number of robots that must be considered for a given environment, as discussed in §6.

Backpropagation of constraints is based on the observation that if an edge exists between two nodes and there are a known set of constraints that would allow a path from the child node to the goal, then the parent node's constraints are the union of the child node's and the edge's. For the same transition discussed above, we have $C(n_{11}) = C(n_{12}) \cup C(n_{11} \to n_{12})$ If multiple paths exist from a given node to the goal, the constraints at that node are the disjunction of those from all edges leaving it. For

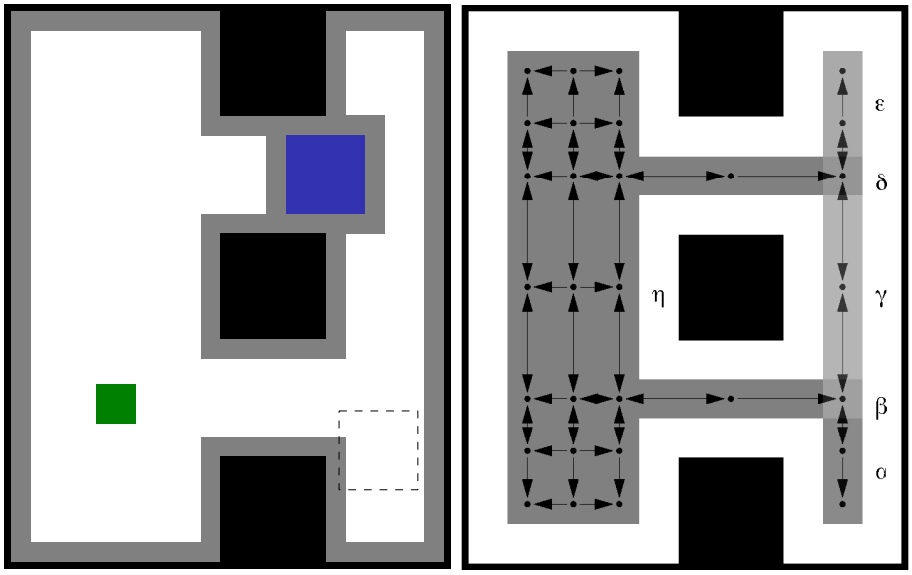


Fig. 3: Environment with topologically distinct paths to the goal, requiring different numbers of robots. Goal is shown as dashed outline; ECs are labeled and different shades of gray.

example, the environment shown in Fig. 3 has two topologically distinct paths from the initial condition to the goal, described here in terms of the ECs that must be traversed:

$$\eta \to \delta \to \gamma \to \beta \to \alpha \qquad \text{(right)}$$
$$\eta \to \beta \to \alpha \qquad \text{(left)}$$

If either path is feasible, then a feasible path exists from $n_\eta \to n_\alpha$, so $C(n_\eta) = (C(n_\delta) \cup C(n_\eta \to n_\delta)) \vee (C(n_\beta) \cup C(n_\eta \to n_\beta))$. Determining whether a feasible path exists requires solving at least one satisfiability problem, which is in general NP-complete, but in practice, efficient solvers exist. Additionally, search efficiency involving the constraints can be improved by preferentially propagating constraints along edges in the FTG that are known to lie on a path connecting the start and goal nodes.

Since the FTG is complete by construction, so long as we use a complete graph search algorithm the resulting planner is also complete. The resulting plan includes an assignment of robots to connected components for each EC that the object passes through. These constraints on robot position can be fed into a multirobot path planner to generate a set of robot paths that are guaranteed to push the object to the goal.

We have discussed a solution to **P2**. This is a special case of **P1**, which asks if any solution exists independent of initial robot assignments. For **P1**, only backpropagation of constraints is used, and it is necessary to check if a satisfying assignment of robots to connected components exists at the start node. Optimizing for object distance or number of robots (**P3**) can be achieved by ordering the FTG traversal based on these metrics.

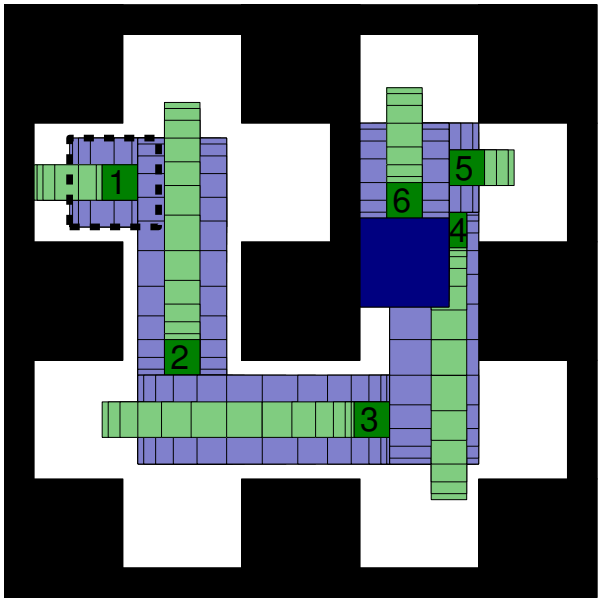


Fig. 4: Example environment for object pushing. We show an object path that would require 6 robots, numbered in the order that they push the object. Intermediate robot and object positions are shown in lighter colors. A dashed square shows the initial object position.

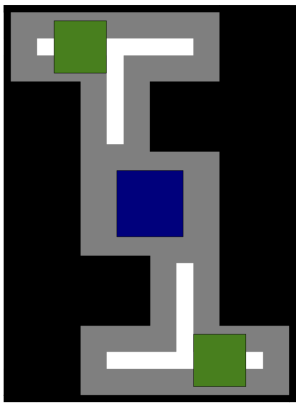


Fig. 5: Environment where using the set of all constraints would give an overestimate of the minimum sufficient robots.

6 The Minimum Sufficient Robots Problem

In this section, we derive a bound on how many robots are required to push the object along any feasible path in a given environment (**P4**). We term this the *minimum sufficient robots* (MSR) problem. The resulting bound applies to the solution found by any planning algorithm. It is of interest for determining how many robots to purchase or deploy and for classifying how challenging a particular environment is for multirobot object manipulation tasks.

An important distinction is that we are considering every feasible path in the environment, not just the path-length-optimal ones. The environment shown in Fig. 5 demonstrates why it is necessary to propagate the constraints through the FTG, rather than simply consider the union of all constraints in the EG. It is not possible for the object to travel from the top half to the bottom half of the environment, so the full set of constraints would lead to an overestimate of how many robots are required. Consider the environment shown in Fig. 4. The object path shown requires six robots to be executed. However, there exists a path between the same initial and final positions that requires only four robots, and there is a path in this environment that requires nine robots. Thus, the MSR bound is the least upper bound to the number of robots required to solve all point-to-point object motions, disregarding the path taken.

The feasible transition graph is designed to propagate constraints throughout the environment, which allows us to find a tight bound on the minimum sufficient number

of robots. Just as in the planner, constraints on a directed FTG edge impute constraints on the edge's parent node. The two problems differ with respect to how constraints are joined when there are multiple edges leading from a node. In the planner, we only require that *a* feasible path exists to the goal; in the MSR problem, we require that *all* paths be feasible. Thus, rather than taking the disjunction of constraints from outgoing edges, we take the conjunction. A solution to the MSR problem can then be found by fully propagating these constraint sets backward through the feasible transition graph until the graph becomes consistent. By consistent, we mean an assignment of constraints to each node that will not change upon any further constraint propagation. In a consistent FTG, every constraint on a node applies to some feasible object path starting at that node.

In order to solve **P4**, constraints must be propagated from every node, not just the goal. After achieving a consistent set of constraints, the minimum sufficient number of robots will be determined by the node whose set of constraints requires the most robots. We have now reformulated the minimum sufficient robots problem as a constrained integer minimization problem with bounded variables. In the worst case, solving this requires a graph search over all possible combinations of variable assignments [17], but the problem structure will allow improvement using heuristics to guide the search.

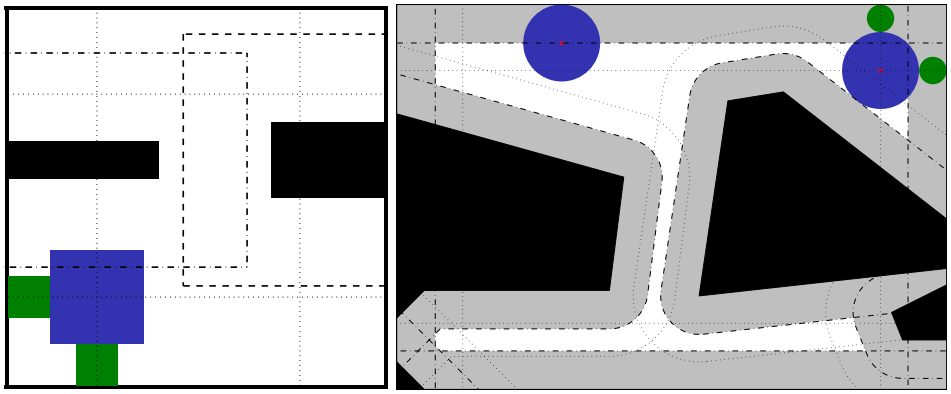
The procedure outlined above is conceptually straightforward but computationally inefficient, as it can require up to $|N_r|$ traversals of the FTG to make the graph consistent. This can be eliminated by preprocessing the FTG to condense it into a directed, acyclic graph whose nodes are the FTG's strongly connected components. The initial constraints on each new node are the union of all constraints in the corresponding strongly connected component. This acyclic graph will only require one additional traversal to propagate all constraints.

7 Implementation

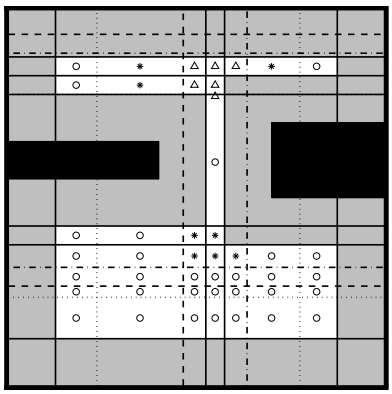
We present implementations of the EG and FTG in two different environments, which were chosen in an attempt to provide as simple an example as possible while still retaining the complex configuration space structure of interest. Both environments allow exact decomposition of Q_M^{free} into ECs; the first also has a completely-specified FTG, whereas we use a probabilistic approach for the second.

Axis-Aligned Box Obstacles All objects in this environment (**O**, M, and **R**) are closed, axis-aligned rectangles. Surface contact, including sliding, is allowed between any pair of objects, but the intersection of their interiors must be empty. The robots **R** may translate freely within their respective connected components of $Q_R^{free}(q_M)$. All motion of the object M is aligned with an axis and is generated by a single robot R_i pushing M, in face-to-face contact. The resulting $M+R_i$ assembly can only move in the direction of M's inward-pointing contact normal. This is the same environment as van den Berg et al. [25] use, but with different constraints on object manipulation.

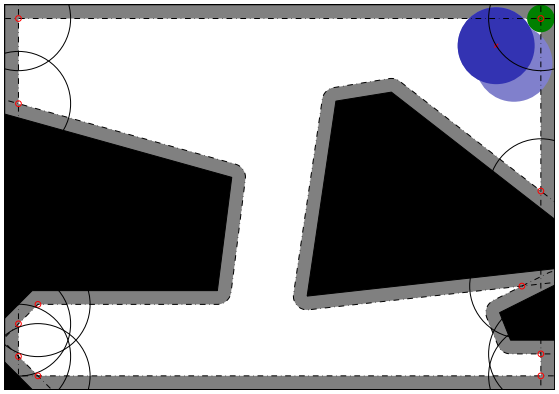
Polygonal Obstacles In this environment, all obstacles **O** are closed polygons, while the object M and robots **R** are circular disks. As in the axis-aligned environment,



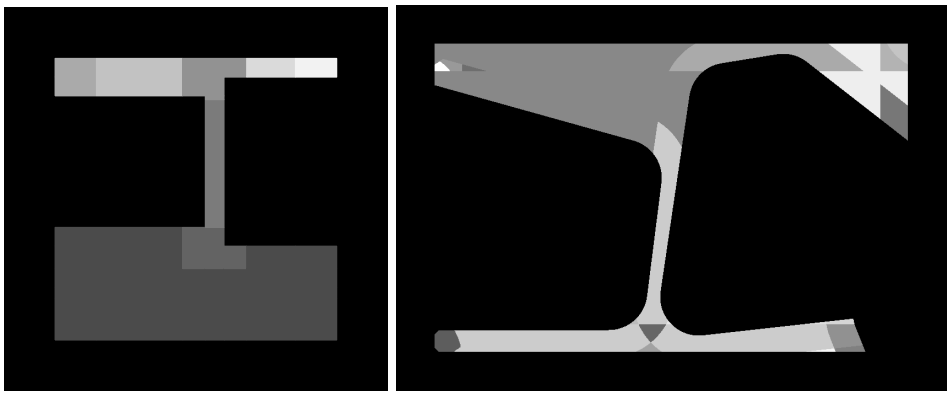
(a) Dotted lines show potential EC boundaries derived from transitions.



(b) Division of Q_M into tiles. Tiles in Q_M^{free} are labeled with $N(q_M)$: $\triangle$ for 3, $*$ for 2, $\circ$ for 1.



(c) The black arcs show potential EC boundaries derived from a disappearing connected component in Q_R^{free}.



(d) Division of Q_M^{free} into ECs. Black indicates C-space obstacle, and each different shade of gray is a different EC.

Fig. 6: Calculation of Equivalence Classes

the robots translate freely within connected components of $Q_R^{free}(q_M)$. Contact among robots, or between the environment and robots or the object, is forbidden. Two robots pushing in tandem are required to generate object motion. For robot–object normals $\hat{n}_i = \frac{q_M - q_{R_i}}{|q_M - q_{R_i}|}$, the possible object motion directions are given by $\hat{v}_M = a\hat{n}_1 + b\hat{n}_2$, for $0 < a, b < 1$.

7.1 Equivalence Graph

There are two types of EC boundaries: those imposed by the boundaries of Q_M^{free} and those created by transitions between ECs. Transitions between ECs correspond to the object "pinching off" or "opening up" a previously (im)passable corridor for the robot (*connectivity*), or to the object's motion causing a connected component of Q_R^{free} to disappear (*existence*). This is because, by definition, any object displacement Δq that results in the object crossing one EC boundary requires that $N(q_M) \neq N(q_M + \Delta q)$. Connectivity changes can only occur when the object's edge is exactly a robot width or height away from an obstacle. For the axis-aligned environment, these boundaries correspond to extending the obstacles by $R_R + 2R_M$. For the polygon world, these boundaries are the configuration space obstacles for a disk with radius $R_R + 2R_M$ (Fig. 6a). In the axis-aligned environment, existence of connected components only changes along the same bounds as the connectivity changes. For the polygon environment, changes in connected component existence will occur when the robot is wedged into a corner, and the EC boundary corresponds to an arc of radius $R_M + R_R$ around the robot's location (Fig. 6c).

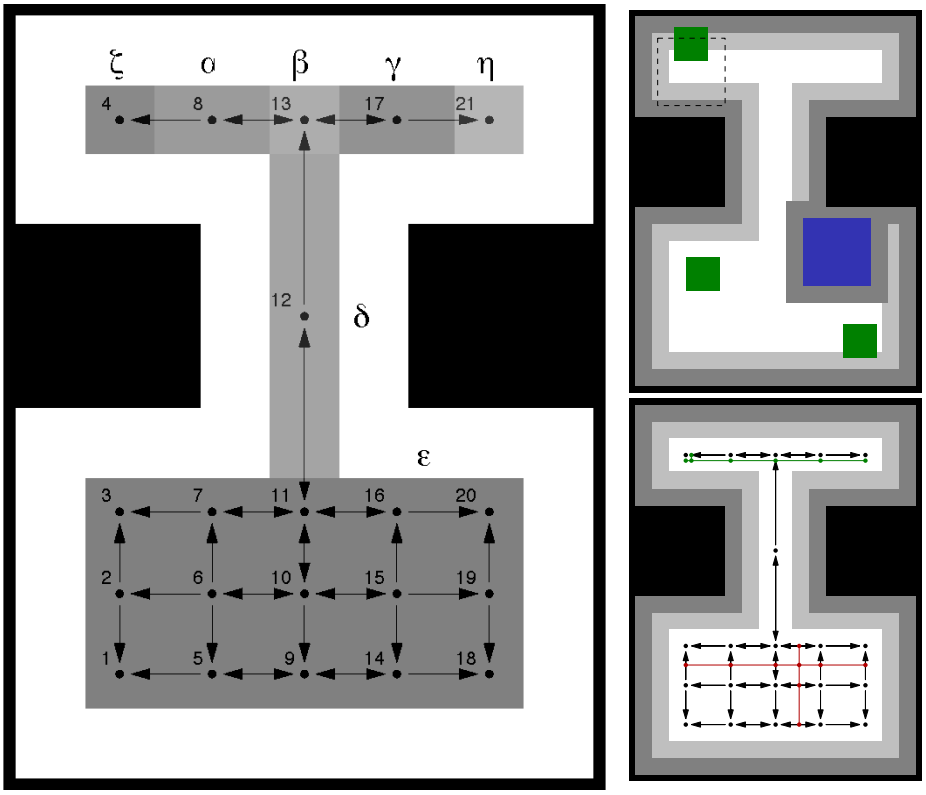
We now have a tiling of the environment, where all q_M in the interior of each tile are guaranteed to result in the same $N(Q_R^{free}(q_M))$. For each tile, we determine the number of connected components in the robots' free space (Fig. 6b). Equivalence classes correspond to the union of neighboring tiles with the same $N(Q)$. Fig. 6d shows exact decompositions of both environments into ECs.

Finally, we need to find the edges of the EG, which correspond to possible object motions that change the connectivity of $Q_R^{free}(q_M)$. In these environments, such edges are simple to compute, as they connect any ECs that share a spatial boundary. They are then labeled with the conservation of robots constraints associated with motion between those ECs. Fig. 2 shows a representation of the EG for the same environment as the left column of Fig. 6.

7.2 Feasible Transition Graph

Axis-Aligned Environment The tiling calculated in §7.1 also captures all changes in possible object motions, as pushing requires a robot to fit behind the object. For an object motion $q_{M1} \rightarrow q_{M2}$, feasibility is calculated by determining if the boundary between a connected component of $Q_R^{free}(q_{M1})$ and the trailing edge of the object's C-obstacle has non-zero length.

In order to represent possible object motions within and along tile boundaries, the FTG's nodes are chosen to be the centroids of each tile, along with mid-points of boundaries and the edges. Note that all locations within a tile will have the same feasible motions as the centroid, so this sampling fully specifies all possible object motions



(a) FTG, subsampled for clarity (only nodes at tile centers are drawn). ECs are shown as different shades of gray, and the labels match those in Fig. 2

(b) Example instance of **P2** (top), and connecting start/goal to the FTG (bottom).

Fig. 7: FTG and planning problem for a simplified version of Fig. 2's environment.

throughout Q_M^{free}. Directed edges are added for any feasible motion between nodes, and labeled with the corresponding constraints on connected component occupancy from the EG. In order to have the required information to plan in the graph, any edge that involves transitioning between ECs is also labeled with the corresponding constraints on connected component occupancy from the EG. Fig. 7a shows a representation of the FTG for a simplifed version of the environment shown in Fig. 2.

Polygonal Environment The purpose of the FTG is to discover feasible motion sequences within Q_M^{free}. We present a general, stochastic method based on Probabilistic Roadmaps (PRMs) [8].

In this application, the PRM randomly samples object configurations from Q_M^{free} and tries to connect them to nearby configurations. An edge E^{FTG} in this roadmap represents a trajectory in the configuration space of the object that obeys the manipulation semantics. Computing these edges requires an inverse manipulation model. In so doing, robots may be placed wherever they are needed to complete the motion. Collision-free

edges are annotated with the number of robots required in each connected component to perform the transfer action. This roadmap describes feasible object motions both within each EC and across EC boundaries.

We note the resemblance of this structure to Multi-Modal PRM [7] and the manipulation planning PRM of Siméon et al. [18]. Each generates a roadmap connecting several manifolds of arbitrary dimension, which are bridged by lower-dimensional intersection manifolds. We show that our formulation of the FTG is probabilistically complete by reducing it in the context of a planning problem (**P2**) into an instance of the Multi-Modal PRM (MM-PRM) of Hauser and Latombe, constructed in the joint configuration space of the object and n robots.

Consider an edge E^{FTG} in the FTG joining two object states, $q_1^{FTG}, q_2^{FTG} \in Q_M^{free}$. We show that this edge is equivalent to edges $E_1^{MM} \dots E_k^{MM}$ in the MM-PRM, with $k \geq 2$, representing a motion connecting $q_1^{MM}, q_2^{MM} \in Q_M^{free} \times Q_R^{free^n}$. We may separately consider *transit* and *transfer* tasks of the robots. In transit tasks (E_1^{MM}), the robots alone move, whereas transfer tasks ($E_2^{MM} \dots E_k^{MM}$) involve manipulation of the object by the robots.

E^{FTG} is annotated with constraints on the number of robots in several connected components, which specify goal states for the robots. By the definition of a connected component, each transit is a motion planning problem that can be solved easily by a standard PRM. For homogeneous robots, the multirobot planning problem can be simplified by selectively permuting goal positions to avoid conflicts [23]. Note that not all robots need to move.

Transfer tasks, whether within or between ECs, are defined by the coordinated motion of the object and some subset of the robot. The motion of the relevant robots is given by the inverse manipulation model. In the case of intra-EC motion ($k = 2$), the other robots do not need to move. For inter-EC motion, $k > 2$ because the transfer edges in the MM-PRM must meet at a point on the boundary between ECs. In this case, the other robots must move to ensure they are in the correct connected component after the transition. Any goal state within the new connected component is an acceptable goal. Again, some robots may not move.

Any edge in the FTG may be mapped to an edge in a connected MM-PRM. Therefore, the probabilistic completeness property of MM-PRMs applies also to this FTG construction. Unlike MM-PRMs, the ECs in which we sample are typically the full dimension of Q_M and the space of edges that cross an EC boundary is likewise of full dimension. Consequently, it is not typically necessary to explicitly sample on the boundary in order to get a connected FTG.

In comparison to building a roadmap directly in the high-dimensional joint configuration space of the object and robots, we can get away with a lower dimensional roadmap by exploiting structure in the problem. Specifically, there is minimal coupling in the motion among the object and robots. The EG allows us to specify goals for the robots in advance without excessive precision. That is, provided that each robot is in the correct EC, detailed positioning is a simple, decoupled motion planning problem.

7.3 Planner

In the axis-aligned environment, simply connecting the points $q_{M,init}, q_{M,final} \in Q_M^{free}$ to the graph does not preserve connectivity. Consider the case of a narrow hallway—if no

motion perpendicular to the hallway is feasible, it is possible that there will be a feasible path between two configurations that can never reach a point in the graph. Instead, we extend the graph to include nodes corresponding to the intersection of both points' coordinates with the all other FTG edges (Fig. 7b, bottom). The polygonal environment is simpler, as the start/goal positions can be connected to the FTG in the same way as the randomly sampled configurations.

8 Discussion and Future Work

In this paper, we propose the Feasible Transition Graph, a representation for multi-robot object-pushing in cluttered environments. This approach enables a user to reason about resource allocation, including how many robots are needed and where they should be positioned while planning motions for the object. We provide complete algorithms for solving these planning problems, and we describe how to construct the FTG for a few simple environments. Our approach exploits the structure of transient independence among robots to construct a much simpler representation than the naive search space comprising the joint configuration space of the object and all robots.

In future work, we plan to consider more general environments, particularly those with a higher-dimensional configuration space. The probabilistic FTG construction approach is already quite general, but we plan to investigate a probabilistically complete construction of the EG for diverse environments as well.

Additionally, there are a number of simple extensions from what we described in detail. First, heterogeneous robots may be handled in one of two ways. If they are of different sizes, then there will be additional EC boundaries corresponding to each new robot radius. If they have different capabilities, then we must introduce additional variables to the number of robots in a given connected component matching that capability. In this way, we could handle planning for robots that must cooperate to push and pull an object.

Acknowledgements

The authors thank Geoffrey Gordon for his incisive comments on early drafts of this work. Laura Lindzey was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

References

[1] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, 4(1):7–27, 1997.

[2] Mark de Berg and Dirk H.P. Gerrits. Computing push plans for disk-shaped robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010.

[3] Keith L Doty and Ronald E Van Aken. Swarm robot materials handling paradigm for a manufacturing workcell. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 778–782. IEEE, 1993.

[4] Jonathan Fink, M. Ani Hsieh, and Vijay Kumar. Multi-robot manipulation via caging in environments with obstacles. In *IEEE International Conference on Robotics and Automation*, 2008.

[5] Russell Gayle, William Moss, Ming C. Lin, and Dinesh Manocha. Multi-robot coordination using generalized social potential fields. In *Proceedings of the IEEE international conference on Robotics and Automation*, 2009. ISBN 978-1-4244-2788-8.

[6] Masafumi Hashimoto, Fuminori Oba, and Toru Eguchi. Dynamic control approach for motion coordination of multiple wheeled mobile robots transporting a single object. In *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 3, pages 1944–1951. IEEE, 1993.

[7] Kris Hauser and Jean-Claude Latombe. Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 29(7):897–915, 2010.

[8] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[9] Yoshihito Koga and Jean-Claude Latombe. On multi-arm manipulation planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1994.

[10] Steven M. LaValle. *Planning Algorithms*. Cambridge press, 2006.

[11] Kevin M. Lynch and Matthew T. Mason. Controllability of pushing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1995.

[12] Matthew T. Mason. *Mechanics of Robotic Manipulation*. MIT press, 2001.

[13] Maja J. Mataric, Martin Nilsson, and Kristian T. Simsarian. Coperative multi-robot box-pushing. In *Proceedings of IROS*, 1995.

[14] G. A. S. Pereira, V. Kumar, and M. F. M. Campos. Decentralized algorithms for multirobot manipulation via caging. *International Journal of Robotics Research*, 2004.

[15] Daniela Rus. Coordinated manipulation of objects in a plane. *Algorithmica*, 19(1-2):129–147, 1997.

[16] Daniela Rus, Bruce Donald, and Jim Jennings. Moving furniture with teams of autonomous robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995.

[17] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003.

[18] Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8): 729–746, 2004.

[19] Reid Simmons, Sanjiv Singh, David Hershberger, Josue Ramos, and Trey Smith. Coordination of heterogeneous robots for large-scale assembly. In *Proceedings of the International Symposium on Experimental Robotics (ISER), Hawaii*, 2000.

[20] Peng Song and Vijay Kumar. A potential field based approach to multi-robot manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[21] Mike Stilman and James J Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, 2(04):479–503, 2005.

[22] Attawith Sudsang, Fred Rothganger, and Jean Ponce. Motion planning for disc-shaped robots pushing a polygonal object in the plane. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[23] Cynthia Sung, Nora Ayanian, and Daniela Rus. Improving the performance of multi-robot systems by task switching. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2999–3006. IEEE, 2013.

[24] Jur van den Berg, Sachin Patil, Jason Sewall, Dinesh Manocha, and Ming Lin. Interactive navigation of multiple agents in crowded environments. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, 2008. ISBN 978-1-59593-983-8. doi: http://doi.acm.org/10.1145/1342250.1342272.

[25] Jur van den Berg, Mike Stilman, James Kuffner, Ming Lin, and Dinesh Manocha. Path planning among movable obstacles: A probabilistically complete approach. In *Algorithmic Foundations of Robotics VIII*, 2008.