Admissible Abstractions for Near-optimal Task and Motion Planning

William Vega-Brown and **Nicholas Roy**
Massachusetts Institute of Technology
{wrvb, nickroy}@mit.edu

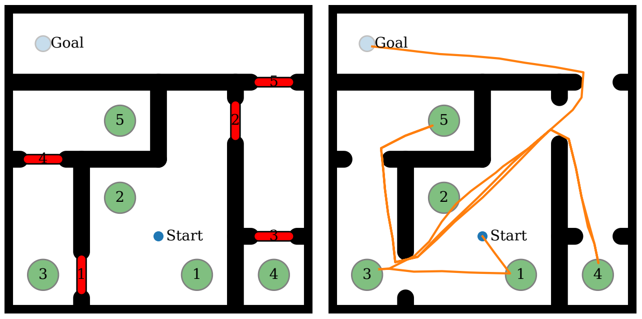
Abstract

We define an admissibility condition for abstractions expressed using angelic semantics and show that these conditions allow us to accelerate planning while preserving the ability to find the optimal motion plan. We then derive admissible abstractions for two motion planning domains with continuous state. We extract upper and lower bounds on the cost of concrete motion plans using local metric and topological properties of the problem domain. These bounds guide the search for a plan while maintaining performance guarantees. We show that abstraction can dramatically reduce the complexity of search relative to a direct motion planner. Using our abstractions, we find near-optimal motion plans in planning problems involving 10^{13} states without using a separate task planner.

1 Introduction

Consider a problem domain like the one shown in figure 1. A holonomic two-dimensional agent is tasked with navigating to a specified goal region as quickly as possible. The path is blocked by doors that can only opened by pressing the appropriate switch. Planning the sequence of switches to toggle requires combinatorial search; deciding if a path exists to each switch requires motion planning. As in many real-world planning domains, such as object manipulation and navigation among movable objects, the combinatorial search and motion planning problems are coupled and cannot be completely separated.

Practical approaches to solving such problems generally use abstraction to reason about the properties of groups of primitive plans simultaneously. For example, the aSyMov system developed by [Cambon *et al.*, 2009] uses a symbolic task planner to compute a lower bound on the number of manipulation operations required to reach a goal state and keeps track of a set of configurations that could be reached following a given sequence of such operations. [Kaelbling and Lozano-Pérez, 2011] use a hierarchy to guide high-level decision making, resolving low-level decisions arbitrarily and trusting in the reversibility of the system to ensure hierarchical completeness. Though the many approaches present in the literature vary widely in how they deal with the interaction between geometric planning and combinatorial search, very few provide any guarantee about the *quality* of the solutions they return. Even optimizing approaches, such as the work of [Wolfe *et al.*, 2010], are generally limited to guarantees of hierarchical optimality. In order to be useful, a planner must generate plans of reasonably low cost, and must do so reasonably quickly; this often requires substantial design effort and domain-specific tuning, and the trade-offs involved are poorly understood.



(a) The door puzzle problem (b) The optimal solution

Figure 1: The door-switch problem, an example task and motion planning domain. A two-dimensional robot must navigate from the start location to a goal location, but the way is obstructed by doors that can only be opened by toggling a corresponding switch. The optimal solution to this problem instance is to toggle the switches in the order $(1, 3, 2, 4, 5)$ and then go to the goal set. Because the size of the configuration space grows exponentially with the number of doors, planning is computationally challenging. Abstraction can render such planning problems tractable.

Angelic semantics [Marthi *et al.*, 2008] provide a way to describe an abstraction that preserves optimality, ensuring that the plans returned have a cost within a user-defined factor of the optimal cost. However, it is unclear precisely what criteria must be met to define an angelic abstraction in a continuous domain. In this paper, we describe sufficient conditions under which an abstraction will preserve the ability to find the optimal motion plan while accelerating planning. We derive abstractions for two continuous planning domains, and we show that using these abstractions can dramatically reduce the complexity of search relative to a direct motion planner. We find near-optimal motion plans in planning problems involving 10^{13} states without using a separate task planner.

2 Problem Formulation

We are interested in planning problems involving some underlying continuous configuration space $\mathcal{X}$, such as the position of a robot or the configuration of its joints. Our task is to find a path through free space that starts in a specified state s_0 and ends in a goal set S_{goal}. This goal set may be specified implicitly, as the set of all states satisfying some constraint.

A path is a continuous map $p : [0, 1] \to \mathcal{X}$. We define a concatenation operator $\circ$ for paths.

$$(p_1 \circ p_2)(t) = \begin{cases} p_1(2t) & \text{if } t \leq \frac{1}{2} \\ p_2(2t-1) & \text{if } \frac{1}{2} < t \leq 1 \end{cases} \tag{1}$$

Let $\mathcal{P}_\mathcal{X}(S, S')$ be the set of all paths starting in $S \subset \mathcal{X}$ and ending in $S' \subset \mathcal{X}$. Let $c : \mathcal{X} \times T\mathcal{X} \to \mathbb{R}_{>0}$ be a cost function, where $T\mathcal{X}$ is the tangent space of $\mathcal{X}$. We can define an associated cost functional $\mathcal{C} : \mathcal{P}_\mathcal{X} \to \mathbb{R}_{\geq 0}$.

$$\mathcal{C}[p] = \int_0^1 c(p(t), \dot{p}(t))\,\mathrm{d}t \tag{2}$$

For example, the arc length of a path p corresponds to a cost function $c(p, \dot{p}) = \|\dot{p}(t)\|$. Because $\mathcal{C}$ is additive, $\mathcal{C}[p_1 \circ p_2] = \mathcal{C}[p_1] + \mathcal{C}[p_2]$. We define the set-valued *optimal cost function* $c^* : 2^\mathcal{X} \times 2^\mathcal{X} \to \mathbb{R}_{\geq 0}$ as

$$c^*(S, S') = \inf\{\mathcal{C}(p) : p \in \mathcal{P}_\mathcal{X}(S, S')\}. \tag{3}$$

We define the ϵ-approximate planning problem as the search for a path $\hat{p} \in \mathcal{P}_\mathcal{X}(\{s_0\}, S_g)$ with cost less than $(1+\epsilon)$ the optimal cost for any $\epsilon \in \mathbb{R}_{\geq 0} \cup \{\infty\}$.

$$\hat{p} \in \{p \in \mathcal{P}_\mathcal{X}(\{s_0\}, S_g) : \mathcal{C}(\hat{p}) \leq (1+\epsilon)c^*(s_0, S_g)\} \tag{4}$$

The case where $\epsilon = \infty$, when we wish to find any feasible path to the goal set, is the problem of *satisficing* planning. The case where $\epsilon = 0$ is optimal planning.

The set $\mathcal{P}_\mathcal{X}(\mathcal{X}, \mathcal{X})$ of all possible paths from all possible start and goal locations is continuous and topologically complex. To simplify planning, we assume we have available a finite set $\mathcal{A}_0$ of *primitive operators*, low-level actions that can be executed in the real world. The problem of constructing such a set of operators in continuous motion planning domains is well studied; in this document, we will assume the set of operators are given by the edges in a probabilistic roadmap (PRM*) [Kavraki *et al.*, 1996; Karaman and Frazzoli, 2011]. That is, we randomly sample a finite set of configurations $\mathcal{V}_n \subset \mathcal{X}$, and for each such configuration v, we define an operator p_v. The operator p_v ensures that the robot will end at the state v if executed from any state in the open ball of radius r_n around v, where $r_n \propto (\log n/n)^{1/d}$ is a radius that increases slowly with the size of the discretization. Any feasible plan can be well-approximated by a sequence of these randomly sampled operators as the number of sampled configurations tends to infinity. For example, we can show that if $\mathcal{A}^*_{0,n}$ is the set of all paths through a PRM* with n sampled configurations, then

$$\lim_{n\to\infty} \inf\{\mathcal{C}[p] : p \in \mathcal{A}^*_{0,n} \cap \mathcal{P}_\mathcal{X}(\{s_0\}, S_g)\} = \inf\{\mathcal{C}[p] : p \in \mathcal{P}_\mathcal{X}(\{s_0\}, S_g)\}. \tag{5}$$

This was proven by [Karaman and Frazzoli, 2011] for the case where the system is subject to analytic differential constraints, and by [Vega-Brown and Roy, 2016] when the system has piecewise-analytic differential constraints (as in object manipulation problems).

Because the set of primitive operators can grow quite large, especially in problems with high-dimensional configuration spaces, a direct search for primitive plans is computationally intractable. Instead, we will use angelic semantics to encode bounds on the cost of large groups of plans. We can use these bounds to plan efficiently while preserving optimality.

3 Angelic Semantics

We define an *abstract operator* $\mathbf{a} \subset \mathcal{P}(\mathcal{X}, \mathcal{X})$ as a set of primitive plans. Because the space of plans is infinite, we define operators implicitly, in terms of constraints on the underlying primitive plans. For example, in a navigation problem, we might define an operator as any primitive plan that remains inside a given set of configuration space and ends in a different set of configuration space.

The concatenation of two operators $\mathbf{a}_1 \circ \mathbf{a}_2$ is an abstract plan containing all possible concatenations of primitive plans in the operators.

$$\mathbf{a}_1 \circ \mathbf{a}_2 = \{p_1 \circ p_2 : p_1 \in a_1, p_2 \in a_2, p_1(1) = p_2(0)\} \tag{6}$$

Concatenations of a well-chosen small set of abstract operators can express very complicated plans in a compact way.

In order to use these abstract operators for planning, we need a way to compare abstract plans. We do this using the *valuation* of an operator or plan. A *valuation* $V[\mathbf{a}]$ for an operator or plan $\mathbf{a}$ is the unique map $V[\mathbf{a}] : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ that takes a pair of states and gives the lowest cost path between the pair.

$$V[\mathbf{a}](s_1, s_2) = \inf\{\mathcal{C}(\sigma) : \sigma \in \mathbf{a}, \sigma(0) = s_1, \sigma(1) = s_2\} \tag{7}$$

Note that if there are no paths in $\mathbf{a}$ linking s_1 and s_2, then $V[\mathbf{a}](s_1, s_2) = \inf \varnothing = \infty$.

Valuations allow us to compare abstract plans without reference to the primitive plans they contain. Given two abstract plans $\mathbf{p}$ and $\mathbf{p}'$, if we can prove that for any pair of states x, x', either $V[\mathbf{p}](x, x') < V[\mathbf{p}'](x, x')$ or $V[\mathbf{p}'](x, x') = \infty$, then either there is a solution to our planning problem in $\mathbf{p}$, or there is no solution in $\mathbf{p}$ or $\mathbf{p}'$. Either way, we do not need to consider any plan in $\mathbf{p}'$; we can prune $\mathbf{p}'$ from our search space. Under such a condition, we say that $\mathbf{p}$ *dominates* $\mathbf{p}'$ and we write $V[\mathbf{p}] \prec V[\mathbf{p}']$. Similarly, if either $V[\mathbf{p}](x, x') \leq V[\mathbf{p}'](x, x')$ or $V[\mathbf{p}'](x, x') = \infty$, then we say that $\mathbf{p}$ *weakly dominates* $\mathbf{p}'$ and we write $V[\mathbf{p}] \preceq V[\mathbf{p}']$.

Unfortunately, determining the valuation of an operator is itself an optimization problem, and one that is not necessarily any easier than the planning problem we are trying to solve. The computational advantage comes from reasoning about bounds on the valuation of an abstract operator. By representing these bounds symbolically, we will be able to reason without reference to the underlying states or plans.

We first define bounds on the valuation of an operator over a set of states.

$$V_L[\mathbf{a}](\mathbf{s}, \mathbf{s}') = \inf\{\inf\{V[\mathbf{a}](x, x') : x' \in \mathbf{s}'\} : x \in \mathbf{s}\} \tag{8}$$

$$V_U[\mathbf{a}](\mathbf{s}, \mathbf{s}') = \sup\{\inf\{V[\mathbf{a}](x, x') : x' \in \mathbf{s}'\} : x \in \mathbf{s}\} \tag{9}$$

A symbolic valuation bound $\hat{V}[\mathbf{a}]$ can be written as a set of tuples $\{(\mathbf{s}, \mathbf{s}', l, u)\}$, where $\mathbf{s}, \mathbf{s}'$ are symbolic states and $l < u \in \mathbb{R}_{\geq 0} \cup \{\infty\}$. A bound $\hat{V}[\mathbf{a}]$ is admissible if

$$\exists(\mathbf{s}, \mathbf{s}', l, u) \in \hat{V}[\mathbf{a}]: \qquad l \leq V_L[\mathbf{a}](\mathbf{s}, \mathbf{s}') \tag{10}$$

$$\forall(\mathbf{s}, \mathbf{s}', l, u) \in \hat{V}[\mathbf{a}]: \qquad u \geq V_U[\mathbf{a}](\mathbf{s}, \mathbf{s}'). \tag{11}$$

In words, a bound $(\mathbf{s}, \mathbf{s}', l, u)$ is admissible if for any state x in $\mathbf{s}$ there exists a plan p ending in some state x' in $\mathbf{s}'$ with cost $c = \mathcal{C}[p]$ bounded above by u and below by l. We can also interpret a symbolic valuation bound $\hat{V}$ as a bound over sets of states.

$$\hat{V}_L[\mathbf{a}](\mathbf{s}, \mathbf{s}') = \inf\{l : (\mathbf{s}_0, \mathbf{s}_1, l, u) \in \hat{V}[\mathbf{a}], \mathbf{s} \cap \mathbf{s}_0 \neq \varnothing, \mathbf{s}' \cap \mathbf{s}_1 \neq \varnothing\} \tag{12}$$

$$\hat{V}_U[\mathbf{a}](\mathbf{s}, \mathbf{s}') = \inf\{u : (\mathbf{s}_0, \mathbf{s}_1, l, u) \in \hat{V}[\mathbf{a}], \mathbf{s} \subseteq \mathbf{s}_0, \mathbf{s}' \subseteq \mathbf{s}_1\}. \tag{13}$$

Note that if $\hat{V}[\mathbf{a}]$ is admissible, then $\hat{V}_L[\mathbf{a}](\mathbf{s}, \mathbf{s}') \leq V_L[\mathbf{a}](\mathbf{s}, \mathbf{s}')$ and $V_U[\mathbf{a}](\mathbf{s}, \mathbf{s}') \leq \hat{V}_U[\mathbf{a}](\mathbf{s}, \mathbf{s}')$ for all abstract state pairs $\mathbf{s}, \mathbf{s}'$.

As we will see in sections 4.1 and 4.2, for many domains we will not need to write down a valuation explicitly. Instead, we can use domain information to make metric computations and generate the necessary elements of a valuation procedurally.

By working with symbolic bounds, we can efficiently compute bounds on the cost of plans consisting of sequences of abstract operators, without reference to a dense discretization of the underlying space of plans. For example, if we have bounds on a plan $\hat{V}[\mathbf{a}]$ and an operator $\hat{V}[\mathbf{a}']$, we can compute a bound $\hat{V}[\mathbf{a} \circ \mathbf{a}']$.

$$\begin{aligned}\hat{V}[\mathbf{a} \circ \mathbf{a}'] = &\{(\mathbf{s}, \mathbf{s}''', l + l', u + u') : (\mathbf{s}, \mathbf{s}', l, u) \in \hat{V}[\mathbf{a}],\\ &(\mathbf{s}'', \mathbf{s}''', l', u') \in \hat{V}[\mathbf{a}'], \mathbf{s}' \subseteq \mathbf{s}''\} \cup\\ &\{(\mathbf{s}, \mathbf{s}''', l + l', u) : (\mathbf{s}, \mathbf{s}', l, u') \in \hat{V}[\mathbf{a}],\\ &(\mathbf{s}'', \mathbf{s}''', l', \infty) \in \hat{V}[\mathbf{a}'], \mathbf{s}' \cap \mathbf{s}'' \neq \varnothing\}\end{aligned} \tag{14}$$

If $\hat{V}[\mathbf{a}]$ and $\hat{V}[\mathbf{a}']$ are admissible, then $\hat{V}[\mathbf{a} \circ \mathbf{a}']$ is admissible as well.

4 Admissible Abstractions

We can use angelic semantics to specify an abstraction that will enable efficient planning. Suppose that $\mathbf{p}, \mathbf{p}'$ are abstract plans, with $\mathbf{p} \subset \mathbf{p}'$. Then $V[\mathbf{p}'] \preceq V[\mathbf{p}]$, since any plan in $\mathbf{p}$ is also in $\mathbf{p}'$—but because $\mathbf{p}$ is a smaller set than $\mathbf{p}'$, our bounds may tighten. If our bounds allow us to conclude that $\hat{V}_U[\mathbf{p}'] \prec \hat{V}_L[\mathbf{p}]$, then we can also conclude that $V[\mathbf{p}' \setminus \mathbf{p}] \prec V[\mathbf{p}']$. We can incrementally construct an increasingly accurate estimate of $V[\mathbf{p}]$ by iteratively considering smaller and smaller subsets of an operator $\mathbf{p}$ and pruning those subsets that cannot contain an optimal plan.

We can make precise the construction of these increasingly fine subsets by introducing a *refinement relation* $\mathcal{R} \subset \mathcal{A}^* \times \mathcal{A}^*$, where $*$ denotes the Kleene closure. The elements of $\mathcal{R}$ are ordered pairs $(\mathbf{p}, \mathbf{p}')$ such that $\mathbf{p}' \subset \mathbf{p}$. We can construct a relation $\mathcal{R}$ by defining several simple operations. First, we define operations $\textsc{Base} : \mathcal{A}^* \to \mathcal{A}^*$, $\textsc{Head} : \mathcal{A}^* \to \mathcal{A}$, and $\textsc{Ext} : \mathcal{A}^* \to \mathcal{A}^*$ that split a plan into three segments so that $\mathbf{p} = \textsc{Base}(\mathbf{p}) \circ \textsc{Head}(\mathbf{p}) \circ \textsc{Ext}(\mathbf{p})$. Second, we define a relation $\bar{\mathcal{R}} \subset \mathcal{A} \times \mathcal{A}^*$. Then $(\mathbf{p}, \mathbf{p}') \in \mathcal{R}$ if and only if $\mathbf{p}' = \textsc{Base}(\mathbf{p}) \circ \mathbf{p}'' \circ \textsc{Ext}(\mathbf{p})$ and $(\textsc{Head}(\mathbf{p}), \mathbf{p}'') \in \bar{\mathcal{R}}$.

We can combine these elements into an *abstraction* over a problem domain $(\mathcal{X}, c, s_0, S_g)$. Formally, an abstraction is a tuple $(\mathcal{S}, \mathcal{A}, \bar{\mathcal{R}}, \hat{V})$, where

- $\mathcal{S}$ is a collection of propositional symbols,
- $\mathcal{A}$ is a collection of operators, including a distinguished top-level operator Act,
- $\bar{\mathcal{R}} \subset \mathcal{A} \times \mathcal{A}^*$ is a refinement relation, and
- $\hat{V}$ is a symbolic valuation bound.

The valuation bound encodes both the cost and the dynamics of our problem domain. The refinement relation structures the space of abstract plans.

Angelic planning algorithms accept an abstraction as an argument in much the same way that the A* search algorithm [Hart *et al.*, 1968] accepts a heuristic. This raises an important question: under what circumstances will an abstraction $(\mathcal{S}, \mathcal{A}, \bar{\mathcal{R}}, \hat{V})$ allow us to find the optimal primitive plan for a domain $(\mathcal{X}, c, s_0, S_g)$, and to prove we have done so? We will generalize the idea of an admissible heuristic to define an *admissible* abstraction.

In fact, two properties suffice.

1. For each abstract operator $\mathbf{a} \in \mathcal{A}$, for each primitive plan p in $\mathbf{a}$, there is a refinement $\mathbf{p}$ of $\mathbf{a}$ such that $p \in \mathbf{p}$, i.e.,
$$\forall \mathbf{a} \in \mathcal{A}, \forall p \in \mathbf{a}, \exists(\mathbf{a}, \mathbf{p}) \in \bar{\mathcal{R}} : p \in \mathbf{p}. \tag{15}$$
2. $\hat{V}$ is admissible, i.e., $\hat{V}_L[\mathbf{p}] \preceq V[\mathbf{p}] \preceq \hat{V}_U[\mathbf{p}]$ for each abstract operator $\mathbf{p} \in \mathcal{A}$.

The first property ensures that we do not "lose track" of any primitive plans while refining a plan. Plans are only removed from consideration when they are deliberately pruned. The second property ensures that if abstract plans $\mathbf{p}, \mathbf{p}' \in P$, where P is a collection of abstract plans, and $\hat{V}_U[\mathbf{p}] \prec \hat{V}_L[\mathbf{p}']$, then no optimal plan is in $\mathbf{p}'$ and thus the best plan in p is also in the set $P' = P \setminus \{\mathbf{p}'\}$. Taken together, these properties ensure that if P' is the result of refining and pruning a collection of plans P, then for every plan in P there is a plan that is no worse in P'. If we start with the set $P_0 = \{\mathrm{Act}\}$, no sequence of refinement and pruning operations will discard an optimal solution. This ensures completeness. To construct planning algorithms, we simply need to choose an order in which to refine and prune, and keep track of bounds to know when we can terminate the search.

Not every admissible abstraction is useful for planning. A good abstraction must be concise: it must be informative enough to enable us to explore the space of plans quickly. In general, designing useful abstractions for a given domain is a complex exercise. In the remainder of this section, we will provide concrete examples of admissible abstractions for a pair of simple continuous planning problems.

4.1 An Abstraction for Navigation

A common problem in robotics is navigating to some specified goal location in a structured environment. Simple heuristics like the Euclidean distance to the goal work well in environments that are cluttered but largely unstructured, where the distance is a good proxy for the true cost. In highly structured environments, however, the Euclidean distance can be quite a bad proxy for cost. Consider the example in figure 3, in which the robot starts just on the other side of a wall from the goal. Using A* with a Euclidean heuristic requires searching almost the entire space.

We can plan more efficiently by taking advantage of structure in the environment. Suppose we have a decomposition of the environment into a finite set of overlapping regions, and we know which regions overlap. Then any plan can be described by the sequence of regions it moves through. We can use this to define an abstraction. Let $\mathcal{S} = \{R_i\}$, where $\cup_i R_i = \mathcal{X}$, and let $\mathcal{A} = \mathcal{A}_0 \cup \{\mathbf{a}_{ij} : R_i \cap R_j \neq \varnothing\} \cup \{\text{Act}\}$, where $p \in \mathbf{a}_{ij}$ if $p(t) \in R_i \forall t \in [0, 1) \wedge p(1) \in R_j$. Define the refinement relation as follows.

$$\begin{aligned}\bar{\mathcal{R}} = \bigcup_{ij} &\{(\text{Act}, \mathbf{a}_{ij} \circ \text{Act})\} \cup \\ &\{(\text{Act}, \mathbf{a}_{ij}) : R_j \cap \mathcal{X}_g \neq \varnothing\} \cup \\ &\{(\mathbf{a}_{ij}, a \circ \mathbf{a}) : a(t) \in R_i \forall t\} \cup \\ &\{(\mathbf{a}_{ij}, a) : a(t) \in R_i \forall t, a(1) \in R_j\}\end{aligned} \tag{16}$$

In practice, we will not iterate over all possible refinements, but will instead use spatial indices like k-D trees and R-trees to find the operators that are valid from a particular state. It is straightforward to show this satisfies the completeness property.

If the cost function is path length, then we can compute bounds using geometric operations. Executing the action $\mathbf{a}_{ij}$ from a state in $R_k \cap R_i$ would incur a cost at least as great as the set distance $\inf\{\|x - x'\| : x \in R_i \cap R_k, x' \in R_i \cap R_j\}$. If the intersections between sets are small and well-separated, this lower bound will be an accurate estimate. This has the effect of heuristically guiding the search towards the next region, allowing us to perform a search in the (small) space of abstract plans rather than the (large) space of primitive plans. The Euclidean heuristic can deal with things like clutter and unstructured obstacles, while the abstraction can take advantage of structure in the environment.

Note that we have made no reference to the shape of the regions, nor even to their connectedness. If regions can be disconnected, abstract operators can have no upper bound, which can lead the search to be inefficient. If we additionally require the regions to be convex, then we can use the Hausdorff distance between sets as an *upper* bound. Executing the action $\mathbf{a}_{ij}$ from a state in $R_k \cap R_i$ would incur a cost no greater than the Hausdorff distance $d_H(R_i \cap R_k, R_i \cap R_j)$, where

$$d_H(X, Y) = \max(\sup_{x \in X} \inf_{y \in Y} \|x - y\|, \sup_{y \in Y} \inf_{x \in X} \|x - y\|). \tag{17}$$

Convexity is quite a strong requirement. In a cluttered environment, a convex representation may need to contain many regions. We can relax the requirement of convexity, and generalize to costs besides path length, by defining ϵ-convexity. A region R is ϵ-convex if

$$\inf_{p \in \mathcal{P}_R(x, x')} \mathcal{C}[p] \leq (1 + \epsilon)\|x - x'\| \, \forall x, x' \in R. \tag{18}$$

ϵ-convex regions are "nearly" convex, because the shortest path connecting any two points is only a bit longer than a straight line. For example, a region cluttered with convex obstacles is $\pi/2$-convex for a point robot.

4.2 An Abstraction for the Door Puzzle

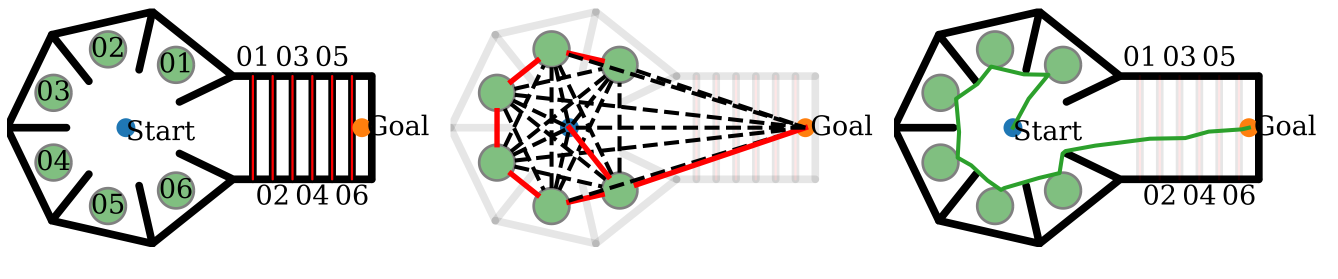
The door puzzle introduced in the introduction combines the motion-planning aspects of navigation with a high-level task planning problem: the choice of which doors to open and in which order. Unlike in the navigation problem, the configuration space for the door problem involves discrete components: $\mathcal{X} \subset \mathbb{R}^2 \times \{0, 1\}^N$, where where N is the number of doors. We use the same region-based abstraction to guide the search for motion plans, and construct a relaxed representation of the effects of toggling switches in PDDL by omitting geometric constraints like collision. Using this representation, we can quickly compute a partial ordering on the sequence of switches that need to be pressed in order to reach the goal. For example, in figure 2, the path to the goal is blocked by six doors. Before we can move towards the goal, we must move to and press each of the six switches. This leaves us with the task of computing a lower bound on the cost to reach and toggle each switch.

We can find such a bound in two steps. First, we construct a directed graph whose vertices are the possible effects of executing each operator, and whose edges have weights that lower bound the cost of executing each operator. This reduces the problem of finding a lower bound to solving a travelling salesperson problem (TSP). While it is believed that solving a TSP requires exponential time, we can compute a lower bound on the cost of the optimal solution by computing a minimum spanning tree of the directed graph—and this is a computation that can be done in polynomial time with standard methods. Although this bound neglects possible interactions between the operators, it is admissible; in fact, it is an admissible special case of the more general (and inadmissible) h_{add} heuristic [Haslum and Geffner, 2000]. We can use this bound to guide the search for a more detailed motion plan.

5 Acyclic Approximate Angelic Search

Next, we present the approximate angelic A* search algorithm (AAA*), an algorithm for near-optimal planning using an admissible abstraction. AAA* is a reformulation and extension of the angelic hierarchical A* algorithm developed by [Marthi *et al.*, 2008]. This algorithm is effectively a best-first forward search over abstract plans. It accepts an abstraction and a positive weight $w \geq 1$, and maintains a queue of plans to expand. The algorithm repeatedly removes from the queue the plan with the lowest lower bound on the cost to reach the goal (lines 8-12). It then constructs a set of *child* plans by selecting one operator from the plan and replacing it with its refinements. Any successor plan that cannot possibly contain an acceptable solution is pruned, while any plan that could contain an acceptable solution is added to the priority queue. The algorithm terminates when we remove a plan from the queue that is dominated by a previously expanded primitive plan.

The primary data structure maintained by our algorithm is a tree. Each node in the tree represents a plan as the concate-



(a) Problem (b) Lower bound (c) Solution

Figure 2: In the problem shown in (a), it is easy to conclude that all $N = 6$ doors must be opened before the robot can reach the goal. However, there are $N! = 720$ possible orders in which we might press the switches. We can bound the cost of any sequence by solving a travelling salesperson problem (b, dotted lines), where the edge costs are the minimal distance the robot must travel to move between switches. Although this is an NP-hard problem, we can compute a lower bound on the cost of a solution in polynomial time by computing a minimum spanning tree (b, solid red line). This allows the planner to quickly find a near-optimal solution (c).

nation of a predecessor plan $\mathbf{p}_-$ and an operator $\mathbf{a}$. Nodes store the following information:

- an abstract operator $\mathbf{a}$,
- the predecessor plan $\mathbf{p}_-$,
- the parent plan, from which this plan was refined,
- the base plan $\textsc{Base}(\mathbf{p})$, which is used in choosing refinements, and
- upper and lower bounds on the valuation of the plan represented by the node.

The root of the tree is a node that contains no operator, predecessor, or base, and whose upper and lower valuation bounds are zero at the start state and infinite elsewhere.

[Marthi *et al.*, 2008] showed this algorithm will return the optimal refinement of the top-level operator Act after a finite number of iterations, provided the lower bound on the cost of every operator is greater than zero. We provide two key modifications. First, instead of expanding the plan with the smallest lower bound, we order the queue with a priority that is no greater than w times the lower bound on the cost a plan. This allows us to exchange optimality for approximate optimality and an accelerated search, in much the same fashion as weighted A* [Pohl, 1970]. Inflating the lower bound may accelerate search by providing a more accurate estimate of the true value of an abstract plan. This is especially true in domains where formulating an abstract plan is difficult: inflating our lower bounds can allow us to focus our search on a single, reasonable abstract plan, rather than considering every possible abstract plan.

Second, the original formulation of angelic A* required strictly positive lower bounds on the cost of any operator. In discrete problems, this is reasonable restriction, but it presents challenges in continuous problems. For example, suppose we have a plan consisting of two operators $\mathbf{a}_{ij} \circ \mathbf{a}_{i'j'}$ from our navigation abstraction. If the destination regions intersect—if $R_j \cap R'_j \neq \varnothing$—then the largest possible lower bound for the valuation of $\mathbf{a}_{i',j'}$ is zero. This phenomenon can lead to a zero-cost cycle: a sequence of operators that can optimistically returns to a given state with zero cost. Even positive cost-cycles are problematic, if the lower bound l on the cost of a cycle is much smaller than the upper bound u: the algorithm can only prune a plan if it executes the cycle $\lceil u/l \rceil$ times. Unfortunately, we cannot simply discard any abstract plan with a cycle: the optimal plan may leave and return to an abstract state if the state is non-convex. Typically, this indicates a poor choice of abstraction, but we can deal with such edge cases while still avoiding cycles with a minor modification to the algorithm.

We define an acyclic plan as any plan $\mathbf{p}$ that cannot be partitioned into two plans $\mathbf{p}_0 \circ \mathbf{p}_1$ such that $\hat{V}_L[\mathbf{p}_0] \preceq \hat{V}_L[\mathbf{p}]$ (algorithm 1, lines 53-58). If when computing the successors of a plan $\mathbf{p}$, we find the extension $\mathbf{p}_{\text{ext}}$ would create a deferred plan when propagated on top of $\textsc{Base}(\text{p})$, we do not add $\mathbf{p} \circ \mathbf{p}_{\text{ext}}$ to the set of successors. Instead, we add $(\textsc{Base}(\mathbf{p}), \mathbf{p}_{\text{ext}})$ to the set of deferred plans (algorithm 1, line 29). When any descendent of $\mathbf{p}$ is expanded, we consider activating any deferred extension of $\mathbf{p}$ by propagating it on top of the descendent plan. If the resulting plan is no longer acyclic, we add it to the set of successors (line 36). This ensures that only acyclic plans will ever be added to the queue of plans without sacrificing completeness.

We state the following theorem without proof; a proof of this theorem, along with proofs of admissibility for the abstractions presented in section 4, are available in an extended version of this paper.[1]

Theorem 1. *If the abstraction $\mathcal{A}$ is admissible and a feasible plan exists, then AAA* returns a sequence of primitive operators with cost no greater than $w \cdot c^*(\{x_s\}, X_g)$ in finite time.*

Corollary 1. *Let $c_n = C[\textsc{Search}(\mathcal{A}_n)]$ be a random variable equal to the cost of the path returned by AAA*. If the set of primitive operators $\mathcal{A}_{0,n}$ is asymptotically optimal (equation 5), then for any $\epsilon > 0$,*

$$\lim_{n\to\infty} \Pr\left(c_n < (1+\epsilon) w \cdot c^*(\{x_s\}, X_g)\right) = 1. \qquad (19)$$

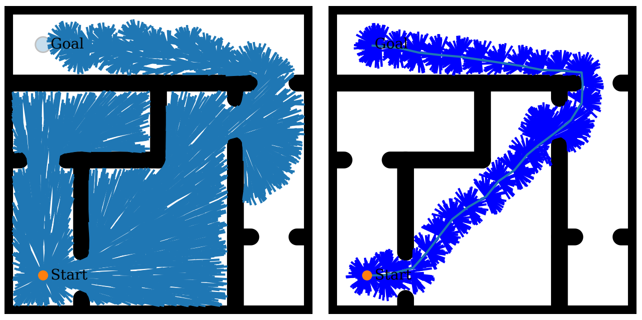
6 Results

We implemented AAA* and the abstractions described in sections 4.1 and 4.2 in the Python programming language. We

[1]https://arxiv.org/abs/1806.00805

Algorithm 1 Approximate Angelic A*

```
function SEARCH(abstraction $(\mathcal{S}, \mathcal{A}, \bar{\mathcal{R}}, \hat{V})$, weight $w$)
    root = $(\varnothing, \varnothing, \varnothing, \{(x_s, x_s, 0, 0)\})$
    $\mathbf{p}^* = \varnothing$
    BOUND($\varnothing$) = $\hat{V}$[ACT]
    $\mathbf{p}_0$ = PROPAGATE(root, [ACT])
    $Q = \{\mathbf{p}_0\}$
    while $|Q| > 0$ do
        $\mathbf{p} = \arg\min\{$KEY$(\mathbf{p}, w) : \mathbf{p} \in Q\}$
        if PRIMITIVE($\mathbf{p}^*$) and $\hat{V}_U[\mathbf{p}^*] \prec \hat{V}_L[\mathbf{p}]$ then
            return $\mathbf{p}^*$
        else
            $Q \leftarrow Q \setminus \{\mathbf{p}\}$
            $S \leftarrow$ SUCCESSORS($\mathbf{p}$)
            for $\mathbf{p}' \in S$ do
                if $\hat{V}_U[\mathbf{p}'](x_0, X_g) < \hat{V}_U[\mathbf{p}^*](x_0, X_g)$ then
                    $\mathbf{p}^* \leftarrow \mathbf{p}'$
            $Q \leftarrow Q \cup S$
    return $\varnothing$
function SUCCESSORS(plan node $\mathbf{p}$)
                                    ▷ $D$ is a global variable, initially set to $\varnothing$.
    POST(BASE($\mathbf{p}$)) = $\{\mathbf{s}' : (\mathbf{s}, \mathbf{s}', l, u) \in \hat{V}[$BASE$(\mathbf{p})]\}$
    $S = \varnothing$
    $\mathbf{a}$ = OPERATOR(HEAD($\mathbf{p}$))
    for $\mathbf{p}' : (\mathbf{a}, \mathbf{p}') \in \bar{\mathcal{R}}, \exists \mathbf{s} \in$ POST(BASE($\mathbf{p}$)) : HEAD($\mathbf{p}'$)$\cap$
$\mathbf{s} \neq \varnothing$ do
        $\mathbf{p}_{\text{ref}} \leftarrow$ PROPAGATE(BASE($\mathbf{p}$), $\mathbf{p}' \circ$ EXT($\mathbf{p}$))
        if $\hat{V}_L[\mathbf{p}_{\text{ref}}](x_s, X_g) < \infty$ then
            if ACYCLIC($\mathbf{p}_{\text{ref}}, \varnothing$) then $S \leftarrow S \cup \{\mathbf{p}_{\text{ref}}\}$
            else
                $D \leftarrow D \cup \{($BASE$(\mathbf{p}),$ EXT$(\mathbf{p}_{\text{ref}}))\}$
    $\mathbf{p}_a \leftarrow \mathbf{p}$
    while BASE(PARENT($\mathbf{p}_a$)) $\neq \varnothing$ do
        $\mathbf{p}_a \leftarrow$ BASE(PARENT($\mathbf{p}_a$))
        for $\mathbf{p}_{\text{ext}} : (\mathbf{p}_a, \mathbf{p}_{\text{ext}}) \in D$ do
            $\mathbf{p}_{\text{ref}} \leftarrow$ PROPAGATE(BASE($\mathbf{p}$), $\mathbf{p}_{\text{ext}}$)
            if ACYCLIC($\mathbf{p}_{\text{ref}}, \varnothing$) and $\hat{V}_L[\mathbf{p}_{\text{ref}}] < \infty$ then
                $S \leftarrow S \cup \{\mathbf{p}_{\text{ref}}\}$
    return $S$
function PROPAGATE(base node $\mathbf{p}$, list $\mathbf{p}_{\text{ext}}$)
    $\mathbf{b} \leftarrow \mathbf{p}$
    while $\mathbf{p}_{\text{ext}}$ is not empty do
        $\mathbf{a} \leftarrow$ POP($\mathbf{p}_{\text{ext}}$)
        if $\mathbf{a}$ is more primitive than OPERATOR($\mathbf{p}$) then
            $\mathbf{b} \leftarrow \mathbf{p}$
        $\mathbf{p} \leftarrow (\mathbf{a}, \mathbf{p}, \mathbf{b}, \hat{V}[\mathbf{p} \circ \mathbf{a}])$
        if $\hat{V}[\mathbf{p}] = \varnothing$ then return $\varnothing$
        else if BOUND($\mathbf{p}_{\text{ext}}$) $\prec \hat{V}_L[\mathbf{p}]$ then return $\varnothing$
        else BOUND($\mathbf{p}_{\text{ext}}$) $\leftarrow$ BOUND($\mathbf{p}_{\text{ext}}$) $\cup \hat{V}[\mathbf{p}]$
    return $A$
function KEY(node $\mathbf{p}$, weight $w \in \mathbb{R}_{\geq 1}$)
    $\mathbf{p}_- \leftarrow$ PREDECESSOR($\mathbf{p}$)
    if $\mathbf{p} = \varnothing$ then return 0
    else return $\min($KEY$(\mathbf{p}_-) + w(\hat{V}_L[\mathbf{p}] - \hat{V}_L[\mathbf{p}_-]), \hat{V}_U[\mathbf{p}])$
function ACYCLIC(plan nodes $\mathbf{p}, \mathbf{p}'$)
    $\mathbf{p}_- \leftarrow$ PREDECESSOR($\mathbf{p}$)
    if $\mathbf{p} = \varnothing$ then return true
    else if $\mathbf{p}' = \varnothing$ then
        return ACYCLIC($\mathbf{p}_-, \varnothing$) $\wedge$ ACYCLIC($\mathbf{p}_-, \mathbf{p}$)
    else return $\neg(\hat{V}_L[\mathbf{p}] \preceq \hat{V}_L[\mathbf{p}'])$ $\wedge$ ACYCLIC($\mathbf{p}_-, \varnothing$)
```



(a) A* (b) Acyclic Angelic A*

Figure 3: The search trees constructed by (a) A* and (b) AAA*. Note that the A* search needs to explore almost the entire space, due to limitations of the Euclidean distance as a heuristic. In contrast, when provided with a decomposition of the world into nearly-convex regions, angelic A* can find a path to the goal while exploring far fewer states. By avoiding plans with cycles, our modified angelic planning algorithm can explore these states while expanding far fewer plans.

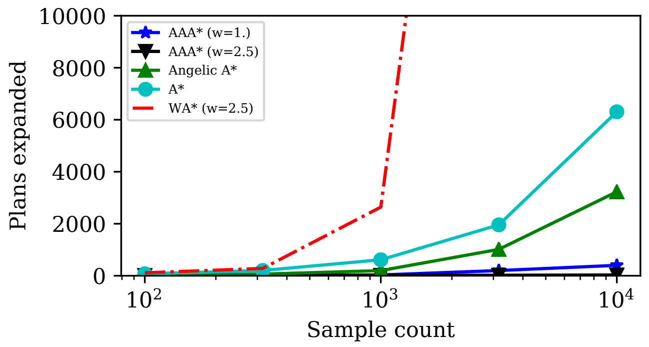
	cost	time	plans	states
A*	33.430	42.119	11807	7948
Angelic A*	33.430	160.256	25770	4758
AAA* (w=1.)	33.430	4.159	706	3068
AAA* (w=2.5)	35.586	0.697	48	1443

Table 1: Quantitative performance on a problem instance in the navigation domain. The discretized state space includes 10^4 sampled configurations. We see that abstraction and approximation result expanding fewer plans and exploring fewer states, yielding a faster search and optimal or nearly optimal results.

then compared the performance of the planner with the original angelic A* search algorithm [Marthi *et al.*, 2008] and with a search without abstraction using A*.

In the navigation domain, we constructed a random discretization with 10^4 states. Examples of the search trees constructed by A* and by AAA* are given in figure 3. By using the abstraction, the algorithm can avoid exploring large parts of the configuration space. Our quantitative results bear this out: using abstraction allows us to reduce the number of states explored by a factor of three and the number of plans considered by several orders of magnitude.

Using abstraction in the door puzzle domain resulted in even larger speedups. Even in easy problem instances with only a few doors, search without abstraction quickly became infeasible (figure 4). Using abstraction reduced the number of states explored by orders of magnitude. However, the unmodified angelic search spent a great deal of time exploring plans with cycles. By deferring these plans, our algorithms were able to reduce the number of plans expanded by an order of magnitude. In fact, only our algorithm was able to solve problem instances with more than ten doors. We were able to find 2-optimal plans for instances with up to 32 doors and 10^4 sampled configurations (corresponding to a discretized state space with approximately 40 trillion states). Unfortunately, software limitations prevented us from experimenting on states with more than 32 doors.



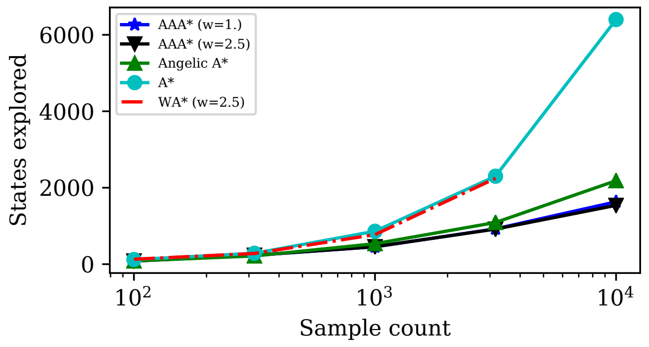


Figure 4: Quantitative evaluation on an easy instance of the door puzzle domain with only two doors. More difficult instances could not be solved by any algorithm considered except AAA*. The abscissa measures the number of randomly sampled states in the discretization of the configuration space. The ordinate axes measure the number plans expanded by each algorithm and the number of distinct configurations explored during search.

7 Related Work

There is a long history of using abstraction to solve robotic planning problems [Nilsson, 1984; Lozano-Pérez *et al.*, 1987]. Many authors [Alami *et al.*, 1990; Siméon *et al.*, 2004] have employed our underlying approach of searching for paths through a graph of configurations connected by feasible motion plans. Practical algorithms often overcome the high computational cost of searching these planning graphs using clever heuristics. For example, aSyMov [Cambon *et al.*, 2009] and FFRob [Garrett *et al.*, 2016] both employ the fast-forward heuristic, augmented with information derived from the geometric and kinematic computation. Like these approaches, our work is built atop a heuristically-guided search; however, angelic semantics allow us to define upper bounds which can be used to prune away abstract plans, and allow for admissible hierarchies of arbitrary depth.

Our definition of abstract plans is closely related to the notion of "plan skeletons" considered by several authors [Erdogan and Stilman, 2013; de Silva *et al.*, 2013; Lozano-Pérez and Kaelbling, 2014]. Plan skeletons fix a sequence of operators but leave continuous parameters undefined. There are many approaches to determining the feasibility of a given skeleton; for example, [Toussaint, 2015] used continuous optimization techniques to search for optimal values of the real-valued variables. [Lozano-Pérez and Kaelbling, 2014] fixed a discretization of the continuous variables then find feasible values by formulating and solving a constraint satisfaction problem. [Lagriffoul *et al.*, 2014] used linear programming to find valid values of the free variables or prove that none exist. The primary difference between our approach and these plan skeletons is the choice of formalism. By defining our abstract operators as implicitly defined *sets* of primitive motion plans, we can reason about plans at varying levels of abstraction in a unified way, which is essential to the generality of our guarantees.

Other approaches to task and motion planning use a representation of geometric information that is amenable to the search techniques of classical AI. For example, [Dornhege *et al.*, 2010] modeled geometric information as predicates to be resolved by solving motion planning problems during the task planning process. More recently, [Ferrer-Mestres *et al.*, 2017] showed that in some domains all geometric information could be represented compactly in planning languages more expressive than PDDL, avoiding the need to make geometric queries during the planning process. Other authors [Erdem *et al.*, 2011; Srivastava *et al.*, 2013; Dantam *et al.*, 2016] used the task planner as a partial or approximate representation of the underlying geometric task, which could be improved during search. For instance, [Erdem *et al.*, 2011] used a high-level task planner to find an optimal task plan, then used a motion planner to attempt to find a kinematically feasible primitive solution to that task plan. If no feasible solution was found, additional kinematic constraints were extracted from the motion planner and provided to the task planner, and the process was repeated.

Many authors have devised planning algorithms tailored to more specific task and motion planning domains. For example, the problem of navigation among movable obstacles has long been of practical interest, and probabilistically complete solutions have been known since 2008 [Stilman and Kuffner, 2008; Nieuwenhuisen *et al.*, 2008]. Planning for non-prehensile manipulation has been addressed by [Dogar and Srinivasa, 2011] and by [Barry *et al.*, 2013]. Our work could provide a new analytical tool with which to study these special classes of problems, and perhaps formulate new algorithms with stronger performance guarantees.

8 Conclusions

We have defined conditions on an abstraction that allow us to accelerate planning while preserving the ability to find an optimal or near-optimal solution to complex motion planning problems. We motivate these conditions by deriving two admissible abstractions and showing they improve the efficiency of search without adversely affecting the quality of the resulting solutions. We view this work as a proof of concept, demonstrating that a good abstraction can render optimal planning feasible even on large problems. The classical planning community has developed several powerful families of admissible heuristics [Haslum and Geffner, 2000]; by reformulating these heuristics to employ angelic abstractions, we may be able to obtain optimal or near-optimal solutions to practical manipulation planning problems.

Acknowledgements

This research was sponsored by Northrop Grumman and by the Robotics Collaborative Technology Alliance (RCTA) of the US Army. Their support is gratefully acknowledged.

References

[Alami *et al.*, 1990] Rachid Alami, Thierry Simeon, and Jean-Paul Laumond. A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps. In *Proceedings of the International Symposium on Robotics Research*, pages 453–463. MIT Press, 1990.

[Barry *et al.*, 2013] Jennifer Barry, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. A hierarchical approach to manipulation with diverse actions. In *International Conference on Robotics and Automation*, pages 1799–1806. IEEE, 2013.

[Cambon *et al.*, 2009] Stephane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126, 2009.

[Dantam *et al.*, 2016] Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems*, pages 1–6, 2016.

[de Silva *et al.*, 2013] Lavindra de Silva, Amit Kumar Pandey, Mamoun Gharbi, and Rachid Alami. Towards combining HTN planning and geometric task planning. *arXiv preprint arXiv:1307.1482*, 2013.

[Dogar and Srinivasa, 2011] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. In *Robotics: Science and Systems*, volume 1, 2011.

[Dornhege *et al.*, 2010] Christian Dornhege, Patrick Eyerich, Thomas Keller, Michael Brenner, and Bernhard Nebel. Integrating task and motion planning using semantic attachments. In *Proceedings of the First AAAI Conference on Bridging the Gap Between Task and Motion Planning*, pages 10–17. AAAI Press, 2010.

[Erdem *et al.*, 2011] Esra Erdem, Kadir Haspalamutgil, Can Palaz, Volkan Patoglu, and Tansel Uras. Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *International Conference on Robotics and Automation*. IEEE, 2011.

[Erdogan and Stilman, 2013] Can Erdogan and Mike Stilman. Planning in constraint space: Automated design of functional structures. In *International Conference on Robotics and Automation (ICRA)*, pages 1807–1812. IEEE, 2013.

[Ferrer-Mestres *et al.*, 2017] Jonathan Ferrer-Mestres, Guillem Francès, and Hector Geffner. Combined task and motion planning as classical AI planning. *arXiv preprint arXiv:1706.06927*, 2017.

[Garrett *et al.*, 2016] Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Ffrob: Leveraging symbolic planning for efficient task and motion planning. *The International Journal of Robotics Research*, page 0278364917739114, 2016.

[Hart *et al.*, 1968] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Haslum and Geffner, 2000] Patrik Haslum and Héctor Geffner. Admissible heuristics for optimal planning. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems (AIPS)*, pages 140–149, 2000.

[Kaelbling and Lozano-Pérez, 2011] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.

[Karaman and Frazzoli, 2011] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

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

[Lagriffoul *et al.*, 2014] Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 2014.

[Lozano-Pérez and Kaelbling, 2014] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *International Conference on Intelligent Robots and Systems*, pages 3684–3691. IEEE, 2014.

[Lozano-Pérez *et al.*, 1987] Tomás Lozano-Pérez, Joseph Jones, Emmanuel Mazer, Patrick O'Donnell, W. Eric L. Grimson, Pierre Tournassoud, and Alain Lanusse. Handey: A robot system that recognizes, plans, and manipulates. In *International Conference on Robotics and Automation*, volume 4, pages 843–849. IEEE, 1987.

[Marthi *et al.*, 2008] Bhaskara Marthi, Stuart Russell, and Jason Wolfe. Angelic hierarchical planning: Optimal and online algorithms. In *International Conference on Automated Planning and Scheduling*, pages 222–231, 2008.

[Nieuwenhuisen *et al.*, 2008] Dennis Nieuwenhuisen, A Frank van der Stappen, and Mark H Overmars. An effective framework for path planning amidst movable obstacles. In *Algorithmic Foundation of Robotics VII*, pages 87–102. Springer, 2008.

[Nilsson, 1984] Nils J Nilsson. Shakey the robot. Technical report, DTIC Document, 1984.

[Pohl, 1970] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial intelligence*, 1(3-4):193–204, 1970.

[Siméon *et al.*, 2004] 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.

[Srivastava *et al.*, 2013] Siddharth Srivastava, Lorenzo Riano, Stuart Russell, and Pieter Abbeel. Using classical planners for tasks with continuous operators in robotics. In *International Conference on Automated Planning and Scheduling*, pages 27–35, 2013.

[Stilman and Kuffner, 2008] Mike Stilman and James Kuffner. Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307, 2008.

[Toussaint, 2015] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *International Joint Conference on Artificial Intelligence*, 2015.

[Vega-Brown and Roy, 2016] William Vega-Brown and Nicholas Roy. Asymptotically optimal planning under piecewise-analytic constraints. In *Workshop on the Algorithmic Foundations of Robotics*, 2016.

[Wolfe *et al.*, 2010] Jason Wolfe, Bhaskara Marthi, and Stuart Russell. Combined task and motion planning for mobile manipulation. In *International Conference on Automated Planning and Scheduling*, pages 254–258, 2010.