Computing a Minimal Set of t-Spanning Motion Primitives for Lattice Planners

Alexander Botros and Stephen L. Smith

***Abstract*— In this paper we consider the problem of computing an optimal set of motion primitives for a lattice planner. The objective we consider is to compute a minimal set of motion primitives that t-span a configuration space lattice. A set of motion primitives t-span a lattice if, given a real number t greater or equal to one, any configuration in the lattice can be reached via a sequence of motion primitives whose cost is no more than t times the cost of the optimal path to that configuration. Determining the smallest set of $t-$spanning motion primitives allows for quick traversal of a state lattice in the context of robotic motion planning, while maintaining a $t-$factor adherence to the theoretically optimal path. While several heuristics exist to determine a t-spanning set of motion primitives, these are presented without guarantees on the size of the set relative to optimal. This paper provides a proof that the minimal $t-$spanning control set problem for a lattice defined over an arbitrary robot configuration space is NP-complete, and presents a compact mixed integer linear programming formulation to compute an optimal t-spanner. We show that solutions obtained by the mixed integer linear program have significantly fewer motion primitives than state of the art heuristic algorithms, and out perform a set of standard primitives used in robotic path planning.**

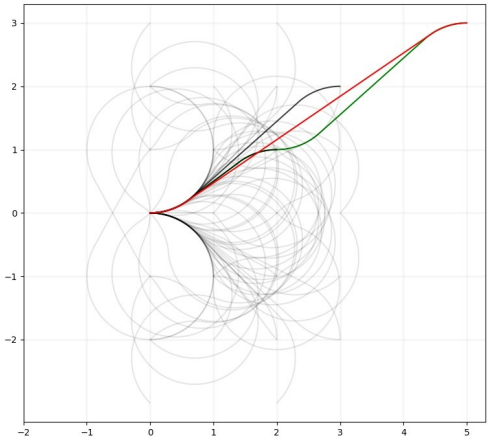


Fig. 1: Example of motion planning using $t-$spanning motion primitives. Vertex $(5, 3, 0)$ is reached optimally from $(0, 0, 0)$ via the red line, and is 1.015-spanned by the light grey motion primitives. The green line represents the path obtained using the motion primitives. Black lines represent motion primitives $(2, 1, 0), (3, 2, 0)$ used to reach $(5, 3, 0)$.

I. Introduction

Sampling based motion planning can typically be split into two methodologies: probabilistic sampling, and deterministic sampling. In probabilistic sampling based motion planning, a set of n samples is randomly selected over a configuration space and connections are made between "close" samples. A shortest path algorithm can then be run on the resulting graph. Examples of probabilistic sampling based algorithms include Probabilistic Road Map (PRM) [1], Rapidly Exploring Random Trees (RRT) [2], or RRT* [3]. Probabilistic sampling is attractive as it avoids explicitly constructing a discretization of the configuration space, and can provide any-time sub-optimal paths. However, theoretical guarantees on the error of generated paths relative to optimal are typically asymptotic in nature due to the randomness of the samples.

In deterministic sampling based algorithms, on the other hand, a uniform discretization of the configuration space, often called a *state lattice*, is explicitly constructed to form the vertices of a graph. In [4], [5], the authors define a state lattice as a graph whose vertices are uniformly distributed over a robot workspace, and whose edges are those tuples (i, j) such that both i and j belong to the vertex set of the lattice, and there exists a controller that brings i to j. The cost of any edge in the lattice is dictated by dynamics of the system in question and the choice of control. The authors of [4] note that any complexity of the system due to its dynamics may be taken into account during the process of lattice construction by pre-computing the cost of each motion in the lattice given the dynamics of the system. This removes the burden of accounting for complex dynamics during the actual process of robotic path planning.

In addition to reducing the complexities involved in robotic path planning, state lattices have also proved versatile in the problems that they can address. For example, adaptations to standard rigid state lattices are widely used in autonomous driving. The authors of [6], [7] demonstrate state lattice adaptations made to account for the structured environments of urban roads.

State lattices can provide theoretical guarantees concerning the cost of proposed paths. Using a connection radius growing logarithmically with the distance between sampled lattice vertices, the authors of [8] provide an upper bound on the error from optimal for a path produced by a deterministic sampling-based road map. The primary drawback with connecting lattice vertices in accordance with a connection radius is the inclusion of redundant neighbours in the search space. For example, for a square lattice in $\mathbb{R}^2$ with Euclidean distances, any-angle dynamics, and a connection radius $r \geq 2$, the vertex $(0, 0)$ will have neighbours $(1, 0)$

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC)

The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, 200 University Ave W, Waterloo ON, Canada, N2L 3G1 (alexander.botros@uwaterloo.ca, stephen.smith@uwaterloo.ca)

and $(2,0)$ while the vertex $(1,0)$ will also have $(2,0)$ as a neighbour. When a shortest path algorithm is run, it will have to consider the path from $(1,0)$ to $(2,0)$ at least twice. As a second example, consider the same lattice, and suppose that the connection radius r is sufficiently large as to ensure that $(0,0)$ and $(4,1)$ are neighbors. Then $(3,1)$ would also neighbour $(0,0)$, and the vertex $(4,1)$ could be reached by first passing through $(3,1)$. The error on this decomposition of the direct path from $(0,0)$ to $(4,1)$ is given by $(\sqrt{3^2+1}+1)(\sqrt{4^2+1})^{-1} = 1.0095$ which may be sufficiently small (0.95%) for the purposes of the user to remove $(4,1)$ as a neighbor of $(0,0)$.

Reducing the number of neighbours of a vertex results in a graph with fewer edges and faster shortest-path computations. These redundancies motivate the following question: given a graph whose vertices are configurations of a mobile robot, and a real number $t \geq 1$, what is the smallest set of edges of the graph that guarantee that the cost-minimizing path from any vertex i to any other vertex j is no more than a factor of t larger than cost of the optimal control between i and j? In other words, what is the smallest set of edges that $t-$span the vertices of the configuration space? This set of edges, called t-spanning motion primitives, correspond to a set of controllers that can be combined to navigate throughout the lattice to within a factor of t of the optimal cost. The notion of $t-$spanning motion primitives is similar to that of graph $t-$spanners first proposed in [9]. An example of a $t-$spanning set of motion primitives is given in Figure 1.

In [10], the authors address the problem of computing a minimal set of t-spanning motion primitives for an arbitrary lattice. We refer to this problem as the minimum t-spanning control set (MTSCS) problem. For a Euclidean graph, attempting to determine such a minimum $t-$spanning set is known to be NP hard (see [11]), and in [10], the authors postulate that the same is true over an arbitrary lattice. The authors of [10] provide a heuristic for the MTSCS problem, which while computationally efficient, does not have any known guarantees on the size of the set relative to optimal.

The main contributions of this paper are twofold. First, we provide a proof that the MTSCS problem is NP-complete for a lattice defined on an arbitrary robot workspace. Because the MTSCS problem is NP-complete, an exact polynomial time algorithm to solve the problem cannot exist (assuming $P \neq NP$). This fact motivates the second contribution of this paper: a mixed integer linear programming (MILP) formulation of the MTSCS problem. This MILP formulation contains only one integer variable for each edge in the lattice. The compact formulation is achieved by establishing that the edges in a minimal lattice spanner must form a directed tree. This MILP formulation constitutes the only non-brute force approach to optimally solving the MTSCS problem. Though this MILP formulation does not scale to very large lattices, it can be solved offline and the solution will hold for any start-goal path planning instance in the same lattice. Several numerical examples are provided in Section V. These examples illustrate that minimal spanners can be computed for complex problems, and the resulting solutions are often significantly smaller than those found by the heuristic in [10]. They also show the quality of paths constructed using the $t-$spanning control set in an A* search of paths in a lattice in the presence of obstacles.

II. Background and Problem Statement

In this section we formalize the notion of state lattices in terms of group theory and then define the minimum t-spanning control set problem.

A. Preliminaries in Group Theory

We begin by summarizing some of the concepts in geometric group theory outlined in [12], modifying some of the notation for our uses.

Consider any subgroup G of the d-dimensional Special Euclidean group $\mathrm{SE}(d)$. By definition, the elements of G are orientation-preserving Euclidean isometries of a rigid body in d-dimensional space. Let s denote the identity element of G. Any isometry i in G represents a possible configuration of a mobile robot. However, i can also be interpreted as a motion taking a mobile robot starting at s to i. As such, motions can be *concatenated* to produce other motions. The group operation of G, denoted $\cdot$ is defined as the left multiplication of elements of G. Thus, for $i, j \in G$, we may define another element $i \cdot j$ that is also in G and represents the concatenation of the isometries i and j. The isometry $i \cdot j$ is the motion that takes s to i by motion i, and then takes i to $i \cdot j$ by motion j.

B. State Lattices

For a mobile robot, let $B_d(x)$ be defined as the set of all points in $\mathbb{R}^d$ occupied by the robot while its center of motion is at $x \in \mathbb{R}^d$.

Definition II.1 (Robot Swath)**.** Consider a mobile robot whose possible configurations are isometries in G. Given $i, j \in G$ and an optimal steering function $u : [0,1] \to \mathbb{R}^d$ such that $u(0) = i, u(1) = j$, we define the *swath* associated with the steering function u between isometries i and j as

$$\mathrm{Swath}_u(i,j) = \{x \in \mathbb{R}^d : \exists \tau \in [0,1], x \in B_d(u(\tau))\}.$$

The set $\mathrm{Swath}_u(i,j)$ represents the set of all vertices in $\mathbb{R}^d$ that the mobile robot will occupy as it traverses a path defined by the steering function from i to j.

Definition II.2 (Valid Concatenation)**.** Consider isometries $i, j, q \in G$ of a mobile robot with optimal steering function u that takes the robot starting at i to j. Suppose that $i \cdot q = j$, and let $W_d^k \subset \mathbb{R}^d$ denote a $d-$dimensional workspace bounded by a ball of radius k. We say that the concatenation $i \cdot q = j$ is *valid* if

$$\mathrm{Swath}_u(i,j) \subseteq W_d^k.$$

If $i \cdot q$ is a valid concatenation, we write $i \oplus q = i \cdot q = j$. If $i \cdot q$ is not valid, we say that $i \oplus q$ is not defined.

Given a workspace, we may determine the set of all valid concatenations of isometries in accordance with the above definition. However, given a set of concatenations V that we wish to declare as valid, we may also define a workspace W_d^k. This is done by assuming that the mobile robot in

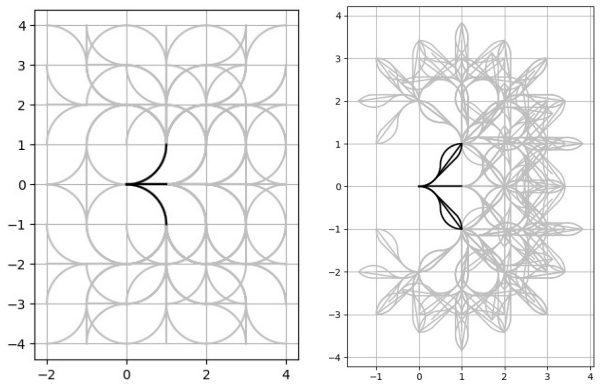


Fig. 2: (Left) Partial lattice with three generators (black) in the group $G = \mathbb{R}^2 \times SO(2)$, cost function given by Dubins' distance for turning radius 1. (Right) Partial lattice generated by six generators (black) in the group $G = \mathbb{R}^2 \times SO(2)$. Cost function c is given by Dubins' distance for a minimum turning radius of 0.5.

question has a controller that takes i to j if and only if $i \cdot q = j$ is a valid concatenation (i.e., $i \oplus q \in V$).

Definition II.3 (Lattice)**.** Let (W_d^k, G, u) denote a workspace, group of isometries, and steering function respectively of a mobile robot. Given a subset $B \subseteq G$, the *lattice* generated by B is defined as

$$L^k(B) = \{j \in G : j = i_1 \oplus i_2 \oplus \cdots \oplus i_m \text{ for some } m \in \mathbb{N}, i_1, ..., i_m \in B\}. \quad (1)$$

The lattice $L^k(B)$ is the set of elements, called *vertices* of G that can be obtained via valid concatenations of elements of B.

Definition II.3 implies that there are two modes of construction of a lattice: lattice vertices may be fixed in $\mathbb{R}^d$ followed by the computation of steering functions from vertex to vertex, or a basic set of control primitives B may be fixed and used to generate a lattice.

Let $c : L^k(B) \to \mathbb{R}_{\geq 0}$ denote a cost function on G. This cost function represents the cost of an optimal control taking s to $q \in L^k(B)$. Assuming that control costs are time invariant, if $i, j \in L^k(B)$ such that $i \cdot q = j$, (i.e., q is the isometry taking vertex i to vertex j) then the cost of the controller taking i to j is given by $c(q)$. In this paper, we assume that

1) $c(q) \geq 0$, for all $q \in L^k(B)$,
2) $c(q) = 0$ implies that $q = s$,
3) if $i, j, k, m, q, r \in L^k(B)$ with $i \oplus m = j$, $j \oplus q = k$, and $i \oplus r = k$, then $c(r) \leq c(m) + c(q)$.

If 1), 2), and 3) hold for a cost function c, we say that c is an *almost-metric*. Note that these assumptions hold for control costs of mobile robots. Observe that c cannot properly be called a metric, as it lacks the symmetry requirement of a metric. Figure 2 illustrates two examples of lattices.

The tuple $(L^k(B), c)$ induces a directed, weighted graph whose *edges* are those tuples $(i, j) \in L^k(B) \times L^k(B)$ such that there exists $q \in L^k(B)$ with $i \oplus q = j$. The *cost* associated with edge (i, j) is $c_{ij} = c(q)$.

Let $E \subseteq L^k(B)$ be any collection of vertices in $L^k(B)$, and suppose that $j \in L^k(E)$. We define a *path* in E to j, denoted $p^E(j)$, as a sequence of m edges (i_r, i_{r+1}) such that $i_{r+1} = i_r \oplus q_r$ for $q_r \in E$ for all $r = 1, \ldots, m-1$, and that takes s to j. The *length* of the path $p^E(j)$ is

$$l = \sum_{r=1}^{m} c_{i_r i_{r+1}}.$$

The *distance* in E to j, denoted $d^E(j)$, is the length of the length-minimizing path in E to j.

For the remainder of this paper, the term path and the notation $p^E(j)$ will refer to a path in E to j of minimal length $d^E(j)$.

C. The Minimum t-Spanning Control Set Problem

We are now ready to introduce the notion of t-reachability of a set of motion primitives in a lattice.

Definition II.4 (t-reachability)**.** Given $(L^k(B), c)$, a set of vertices $E \subseteq L^k(B)$, and a real number $t \geq 1$, a vertex $j \in L^k(B)$ is *t-reachable* from E if

$$d^E(j) \leq t d^B(j) = t c(j).$$

That is, j is $t-$reachable from E if the length of a shortest path $p^E(j)$ in E to j is no more than t times the length of the shortest path from s to j in $L^k(B)$. If every $y \in L^k(B)$ is $t-$reachable from E, we say that E *t-spans* the lattice $L^k(B)$.

The Minimum $t-$Spanning Control Set problem is formulated as follows.

Problem II.5 (Minimum $t-$spanning Control Set Problem)**.**
Input: A tuple $(L^k(B), c)$, and a real number $t \geq 1$.
Output: A set $E \subseteq L^k(B)$ of minimal size that $t-$spans $L^k(B)$.

Observe that E is a set of t-spanning motion primitives for a mobile robot whose configuration space is given by $L^k(B)$. That is, any configuration in the configurations space $L^k(B)$ may be decomposed into a sequence (path) of motions in E such that the cost of any decomposition is no more than a factor of t larger than the cost of the configuration.

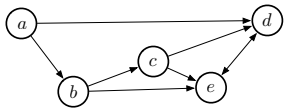
In the remainder of the paper we establish the hardness of the problem and then present a mixed integer linear Programming (MILP) formulation of Problem II.5.

III. Hardness of Computing MTSCs

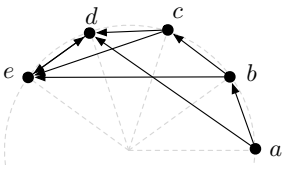
In this section, we begin by characterizing the hardness of Problem II.5, which serves to motivate the MILP formulation presented in the next section.

Theorem III.1. *Problem II.5 is NP-hard.*

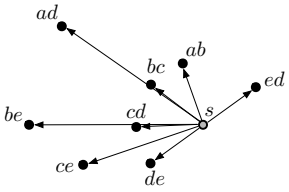
Proof. To show that Problem II.5 is NP-hard, we will construct a reduction from a metric graph $t-$spanner problem. This problem is known to be NP-hard (see [13], [11]). Given a directed graph $G = (V_G, E_G, w)$ with vertex set V_G, edge set E_G, edge weights w, and a value $t \geq 1$, we construct a lattice $L^k(B)$ in the underlying group $\mathbb{R}^2$ with almost-metric cost function c and a real number $t' \geq 1$ that constitutes an instance of Problem II.5.



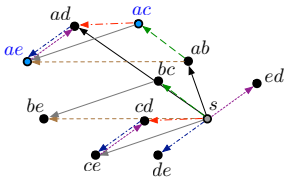
(a) Graph t-spanner input.



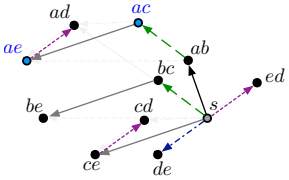
(b) Arrangement with non-equal edge vectors.



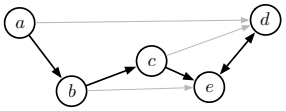
(c) Lattice generating set B.



(d) Complete lattice $L^k(B)$.



(e) Lattice $t-$spanner Q'.



(f) Graph t-spanner Q.

Fig. 3: Illustration of the steps in the reduction from metric graph t-spanner to Problem II.5.

We begin by arranging the graph G in the plane. Vertices $a \in V_G$ will be represented as points in $a \in \mathbb{R}^2$, and edges $(a, b) \in E_G$ will correspond to vectors connecting a and b. The arrangement is done in such a way as to ensure that no two edge vectors are equal and can be performed in time polynomial in the size of the graph.

To construct the arrangement, partition the upper half of the unit circle centered at the origin into $|V_G|$ equally spaced wedges, and place one vertex at the corner of each wedge starting at $(1, 0)$. For each edge $(a, b) \in E_G$, add a vector from a to b in the plane (see Figure 3b). Observe that no two edge vectors are equal. Indeed, each edge vector corresponds to a non-diametral secant line of the unit circle. A circle in $\mathbb{R}^2$ possesses exactly two equal non-diametral secant lines appearing as mirror images about a diameter. Therefore, two equal non-diametral secant lines cannot both occupy the upper half of the unit circle.

We next construct the lattice in the underlying group $\mathbb{R}^2$ from the arrangement. This is accomplished by first attaching each of the edge vectors to the origin. For each edge vector (a, b), declare a vertex ab in the lattice generating set B located at the tip of the edge vector (see Figure 3c). Observe that no two vertices in B are co-located because no two edge vectors are equal. Next, we construct the remaining lattice vertices and edges. For each vertex $ab \in B$, determine the out edges in G of the vertex b, say $(b, c), (b, e)$. Concatenate (by vector addition) the vertices in B corresponding with each out edge (say, bc, be) to the vertex ab. The result of the concatenation is $ab \oplus bc = ac$. If the vertex ac already exists in the lattice, proceed to the next out edge of b. Otherwise, declare vertex $ac \in L^k(B)$. The construction of the lattice is illustrated in Figure 3d.

Finally, to each $ab \in L^k(B)$, define the cost $c(ab)$ as the cost of the minimal path in G from a to b. Observe that this cost is not associated with the Euclidean norm of any vector used in the construction of $L^k(B)$. Observe further, that in constructing the lattice $L^k(B)$, we have implicitly defined a workspace W_2^k that admits concatenations $ab \cdot bc$ if and only if $(a, b), (b, c) \in E_G$, and admits $s \cdot ab$ if and only if $(a, b) \in E_G$.

To prove the correctness of the reduction we show that if Q' is a MTSCS of $L^k(B)$ (see Figure 3e), then there exists a minimal $t-$spanner of G of equal size. Observe that if Q' is a minimal set of $t-$spanning vertices of $L^k(B)$, and $ab \in Q'$, then $(a, b) \in E_G$. Indeed, if $(a, b) \notin E_G$, then no path in $L^k(B)$ can involve a concatenation of any vertex in $L^k(B)$ with ab by the definition of the workspace. As such, ab will not appear in any minimal set of $t-$spanning vertices.

Observe further, that any path $\{u_j, u_{j+1}\}_{j=1}^m$ of m edges in E_G corresponds uniquely to a path of vertices $\{u_j u_{j+1}\}_{j=1}^m$ in $L^k(B)$ and vice verse. Moreover, the total cost of the path in G is equal to the total cost of the path in $L^k(B)$.

Thus, the vertices in Q' correspond to edges in G, and any path of vertices in Q' corresponds to a path of edges in G of equal cost. Therefore, Q' induces a set of edges $Q \subseteq E_G$ such that $(a, b) \in Q \iff ab \in Q'$, and where Q is a $t-$spanner of G. Therefore, the minimal $t-$spanner G cannot have size greater than $|Q'|$. Moreover, it follows that any set Q that is a minimal $t-$spanner of G induces a set Q' of equal size comprised of vertices that $t-$span $L^k(B)$. Therefore, the minimal set of $t-$spanning motion primitives cannot have size larger than Q. It follows then that $|Q'| = |Q|$ and a solution to our constructed instance of Problem II.5 provides a minimal t-spanning set of edges Q for the metric graph t-spanner problem. This is summarized in Figures 3e, and 3f.

□

IV. Computing a MTSCS

In light of Theorem III.1, and assuming that $P \neq NP$, an efficient algorithm to exactly solve Problem II.5 does not exist. However, in what follows we formulate a MILP Problem II.5 that uses one just integer variable for each edge in $L^k(B)$.

A. Properties of Minimal Spanning Control Sets

For any vertex $q \in L^k(B)$, let

$$S_q = \{(i, j) \ : \ i, j \in L^k(B), \ i \oplus q = j\}.$$

That is, S_q is the set of all pairs (i, j) such that q takes i to j and $i \cdot q$ is a valid concatenation. The set of all edges in the graph $L^k(B)$ is given by

$$\mathcal{E} = \bigcup_{q \in L^k(B) - \{s\}} S_q.$$

A naive approach to developing a MILP would be to define $|L^k(B)| \cdot |\mathcal{E}|$ integer variables x_{qij}, each which takes the value 0 if $(i, j) \in S_q$ and $q \notin E$, and value 1 if $q \in E$. However,

the number of required integer variables in the MILP can be reduced to $|\mathcal{E}|$ via a graph theoretic approach to Problem II.5. This approach is motivated by the following definition and lemma:

Definition IV.1 (Arborescence)**.** In accordance with Theorem 2.5 of [14], a graph T with a vertex s is an arborescence rooted at s if every vertex in T is reachable from s, but deleting any edge in T destroys this property.

Lemma IV.2. *Let $E \subseteq L^k(B)$ be a solution to Problem 1. We construct a graph $T = (V_T, E_T)$ whose vertices V_T are those vertices in $L^k(B)$, and whose edges E_T are defined as follows: let*

$$T' = \bigcup_{i \in L^k(B) - \{s\}} p^E(i).$$

For each $i \in L^k(B) - \{s\}$, if T' contains two paths p_1, p_2 to i, determine the last common vertex j in paths p_1, p_2, and delete the edge in p_2 whose endpoint is j from T'. Let the remaining edges be the set E_T. Then T is an arborescence rooted at s, and for each $i \in L^k(B)$, the value $d^E(i)$ is the length of the path in T to i.

Proof. Observe that $L^k(E) = L^k(B)$ since E solves Problem II.5. Therefore, if j is a vertex in T, then it must be reachable from s. Observe further, that for each $j \in V_T$, there exists a unique path in T to j. Indeed, if there were two paths p_1, p_2 of edges in E_T to j, then p_1, p_2 would be paths in the edge set T' because $E_T \subseteq T'$. However, upon construction of E_T from T', an edge from p_2 would be removed, implying that p_2 could not be a path in E_T to j.

Suppose that an edge (i, j) is removed from E_T. Then by the definition of T, there must exist some $r \in V_T$ with (i, j) on the unique path in T to r. Therefore, if (i, j) is removed from E_T, then there is no path of edges in $E_T - \{(i, j)\}$ from s to r, which implies that r is not reachable from s. Therefore, T is an arborescence.

It will now be shown that the length of the path in T to any vertex $i \in V_T$ is $d^E(i)$. Recall that $p^E(i)$ is defined as a path of minimal length in E to i, and the length of this path is $d^E(i)$. Suppose that edge (q, i) is deleted from T' during the construction of E_T. Then it must hold that (q, i) was on a minimal path p_2 to i of length $d^E(i)$. Moreover, it must hold that there exists another path p_1 such that $(q, i) \notin p_1$ whose length is also $d^E(i)$ (otherwise, i would not be a common endpoint to two paths). Therefore, when (q, i) is deleted from T', the minimal distance in E_T to i is still $d^E(i)$, and length of any path containing i remains unchanged. □

Lemma IV.2 implies that if E is a minimum $t-$spanning control set of $L^k(B)$, then there is a corresponding arborescence T whose vertices are those vertices of $L^k(B)$, whose edges (i, j) are members of S_q for some $q \in E$, and in which the cost of the path from s to any vertex $i \in T$ is no more than tc_i.

Suppose now that $|L^k(B)| = n$ and that all the vertices are enumerated as $1, 2, ..., n$ with $s = 1$. For any set $E \subseteq L^k(B)$ such that $L^k(B) \subseteq L^k(E)$, define $n - 1$ decision variables $y_q, q = 2, ..., n$ as

$$y_q = \begin{cases} 1, & \text{if } q \in E \\ 0, & \text{otherwise.} \end{cases}$$

For each $(i, j) \in \mathcal{E}$, let

$$x_{ij} = \begin{cases} 1 & \text{if } (i, j) \in T \\ 0 & \text{otherwise.} \end{cases}$$

Let z_i denote the length of the path in the tree T to vertex i for any $i \in L^k(B)$, and let $L' = L^k(B) - \{s\}$.

B. High-Level Description of Optimization

We begin by providing an high-level description of the objective and problem constraints, followed by a precise MILP formulation. To solve Problem II.5 our objective is to minimize $|E|$ subject to the following constraints:

Usable Edge Criteria: For any $q \in L'$, if $y_q = 1$, then $q \in E$. Therefore, for any $(i, j) \in S_q$, the variable x_{ij} can take either the value 0 or 1. On the other hand, if $y_q = 0$, then $x_{ij} = 0$ for all $(i, j) \in S_q$, implying that (i, j) may not appear in T.

Cost Continuity Criteria: If $x_{ij} = 1$, for any $(i, j) \in \mathcal{E}$, then the path in the arborescence T from s to j contains the vertex i. Therefore, it must hold that

$$z_j = z_i + c_{ij}.$$

$t-$Spanning Criteria: The length of the path in T to any vertex $j \in L'$ can be no more than t times the length of the direct edge from s to j. That is,

$$z_j \leq tc_j, \ \forall j \in L'.$$

Arborescence Criteria: The set T must be an arborescence.

C. MILP Formulation

The constraints listed above can be encoded as the following MILP.

$$\min \sum_{q=2}^{n} y_q \tag{2a}$$

$$s.t. \tag{2b}$$

$$x_{ij} - y_q \leq 0, \qquad \forall (i, j) \in S_q, \ \forall p \in L' \tag{2c}$$

$$z_i + c_{ij} - z_j \leq M_{ij}(1 - x_{ij}), \qquad \forall (i, j) \in \mathcal{E} \tag{2d}$$

$$z_j \leq tc_j, \qquad \forall j \in L' \tag{2e}$$

$$\sum_{i \in L'} x_{ij} = 1, \qquad \forall j \in L' \tag{2f}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall (i, j) \in \mathcal{E} \tag{2g}$$

$$y_q \in [0, 1], \qquad \forall p \in L', \tag{2h}$$

where $M_{ij} = tc_i + c_{ij} - c_j$. The constraints of (2) are explained as follows.

Constraint (2c): If $q \notin E$, then $y_q = 0$ by definition. Therefore, (2c) requires that $x_{ij} = 0$ for all $(i, j) \in S_q$. Alternatively, if $q \in L'$, then $y_q = 1$, and x_{ij} is free to take

values 1 or 0 for any $(i,j) \in S_q$. Thus constraint (2c) is equivalent to the Usable Edge Criteria.

Constraint (2d): Constraint (2d) encodes Cost Continuity Criteria. It takes a similar form to [15, Equation (2.7)]. Begin by noting that $M_{ij} \geq 0$ for all $(i,j) \in E$. Indeed, for any $t \geq 1$,

$$M_{ij} \geq c_i + c_{ij} - c_j,$$

and $c_i + c_{ij} \geq c_j$ by the definition of an almost-metric. Replacing the definition of M_{ij} in (2d) yields

$$z_i + c_{ij} - z_j \leq (tc_i + c_{ij} - c_j)(1 - x_{ij}). \tag{3}$$

If $x_{ij} = 1$, then (3) reduces to $z_j \geq z_i + c_{ij}$. If, however, $x_{ij} = 0$, then (3) reduces to $z_i - z_j \leq tc_i - c_j$ which holds trivially by constraint (2e) and by noting that $z_j \geq c_j, \forall j \in L$ by the definition of an almost-metric.

Constraint (2f): Constraint (2f) together with constraint (2d) yield the Arborescence Criteria. Indeed, by Theorem 2.5 of [14], T is an arborescence rooted at s if every vertex in T other than s has exactly one incoming edge, and T contains no cycles. The constraint (2f) ensures that every vertex in L', which is the set of all vertices in T other than s, has exactly one incoming edge, while constraint (2d) ensures that T has no cycles. To see why this is true, suppose that a cycle existed in T, and that this cycle contained vertex $i \in L'$. Suppose that this cycle is represented as

$$i \to j \to \cdots \to k \to i.$$

Note that (2d) and condition 2 of the definition of an almost-metric imply that $z_i < z_j$ for any $(i,j) \in \mathcal{E}$. Therefore,

$$z_i < z_j < \cdots < z_k < z_i,$$

which is a contradiction.

Constraint (2h): The variable y_q is a decision variable, and therefore should take values in $\{0,1\}$ for all $q \in L'$. However, the integrality constraint on y_q may be relaxed to (2h). Indeed, suppose that for any $q \in L'$, there exists an edge $(i,j) \in S_q$ such that $x_{ij} = 1$. Then, by (2c), $y_q \geq 1$ which implies by (2h), that $y_q = 1$. If, on the other hand, there does not exist $(i,j) \in S_q$ with $x_{ij} = 1$, then y_q is free to take values in $[0,1]$. Therefore, $y_q = 0$ as the objective function seeks to minimize the sum of y_q over q.

Observe that the NP-hardness of Problem II.5 is proved in Theorem III.1, while a reduction from Problem II.5 to an MILP is given in 2. Therefore, a reduction of the decision version (does there exist an MTSCS of specified size?) can be constructed both from and to NP-complete problems, which implies that the decision version of Problem II.5 is NP-complete.

In the next section, the importance of minimal $t-$spanning control sets to the field of motion planning is illustrated by way of numerical examples.

V. Simulation Results

We implemented the MILP presented in (2) in Python 3.6, and solved using Gurobi [16]. In this section, we begin by comparing the size of the set of $t-$spanning motion primitives determined by (2) with those obtained from the sub-optimal algorithm presented in [10]. We also compare paths generated using primitives computed here with those standard primitives appearing in [17]. We conclude with an brief investigation of motion primitives in grid-based path planning.

	$k=3$		$k=4$		$k=7$	
	$\lvert E^*\rvert$	$\lvert E_P\rvert$	$\lvert E^*\rvert$	$\lvert E_P\rvert$	$\lvert E^*\rvert$	$\lvert E_P\rvert$
$R=0.5$						
$t=1.01$	70	70	92	94	124	137
$t=1.5$	9	9	9	9	9	9
$t=3$	6	6	6	6	6	6
$R=2$						
$t=1.01$	75	75	90	92	128	132
$t=1.5$	12	20	13	19	11	19
$t=3$	7	7	10	10	10	11
$R=4$						
$t=1.01$	69	69	102	102	223	231
$t=1.5$	16	16	16	24	19	40
$t=3$	3	5	7	7	13	14

TABLE I: Results for lattice L_1.

A. Comparison to Existing Primitive Generators

The first example we consider is the lattice

$$L_1 = (\mathbb{Z}^2 \cap [0,k] \times [-k,k]) \times \{0, \pi/2, \pi, 3\pi/2\}.$$

The lattice L_1 is generated by the set $B = \{(1,0,0),(1,1,\pi/2),(1,-1,3\pi/2)\}$. The workspace W_d^k is defined as $[0,k] \times [-k,k]$. The mobile robot is assumed to be a single point in $\mathbb{R}^2$, and swaths and costs are defined by Dubins' paths and path lengths, respectively.

For the lattice L_1, the size of the MTSCS, E^*, for varying minimal turning radii R, and values of t and k were computed using the MILP in (2). The sizes of these control sets were compared with those sets (denoted E_P) obtained by employing the heuristic algorithm proposed in [10]. Table I summarizes the findings.

The error $|E_P| - |E^*|$ can be as low as 0 in some cases. However, for certain values of R, t, k, the error can be larger than $|E^*|$. For example, when $t = 1.5, R = 4, k = 7$, the solution to the MILP in (2) is a set E^* whose size is less than half the size of E_P. This will increase the speed of any path planning algorithm that uses the control set E^*.

For the second example we consider the lattice

$$L_2 = (\mathbb{Z}^2 \cap [0,k] \times [-k,k]) \times \{i\frac{\pi}{4}\}_{i=0}^{7}.$$

Observe that L_2 is the eight heading (cardinal and ordinal) version of L_1. Table II summarizes the findings for L_2.

Observe that for $k = 3, R = 4, t = 3$, the size of the set E^* is less than $1/3$ times that of E_P. Tables I, and II imply that there are several instances of mobile robot and desired configurations of the robot for which $|E^*|$ is several times smaller than that of $|E_P|$. Moreover, because it is not obvious, given an instance of problem II.5, when the error $|E^*| - |E_P|$ will be small, it is not a trivial matter to avoid such instances. The runtime to compute each MILP-based

	$k=3$		$k=4$	
	$\lvert E^*\rvert$	$\lvert E_P\rvert$	$\lvert E^*\rvert$	$\lvert E_P\rvert$
$R=0.5$				
$t=1.01$	154	154	196	198
$t=1.5$	19	19	19	19
$t=3$	10	12	10	12
$R=2$				
$t=1.01$	159	159	214	222
$t=1.5$	34	50	31	49
$t=3$	15	22	19	23
$R=4$				
$t=1.01$	147	147	226	232
$t=1.5$	44	44	50	68
$t=3$	5	18	11	20

TABLE II: Results for lattice L_2.

spanning set in Tables I, II ranged from a few seconds to on the order of an hour using a computer with an Intel®Core™ i7-6700 CPU @ 3.40GHz × 8 and 16 GB of memory.

B. Path Planning Comparison

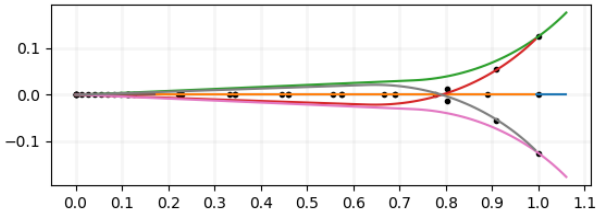
A standard set of motion primitives can be found in the ROS package SBPL [17]. For a minimum turning radius $R = 0.5$, 8 headings (cardinal and ordinal), and cost function given by the Dubins' distance, 40 start-goal path planning problems were created. These problems are comprised of a randomly generated goal location and set of obstacles. Each problem was solved using the same A* implementation. This A* algorithm operates by expanding search nodes starting at $s = (0, 0, 0)$. The neighbors of each search node are are determined by concatenating the node with each of the primitives. A concatenation $i \cdot j$ is deemed valid if and only if the x and y coordinates of $i \cdot j$ are both integers. The primitives used are illustrated in Figure 4. Figure 4a illustrates the 8 primitives proposed in [17] (note that some primitives are not visible as they are short). The 33 primitives in 4b were obtained by solving the MILP in (2) for the lattice $L^k(B)$ generated by

$$\begin{aligned} B_{\text{MILP}} = \{&(1,0,0),(1,1,0),(1,-1,0),(1,1,\pi/4),\\ &(1,-1,7\pi/4),(1,1,\pi/2),(1,-1,3\pi/2),\\ &(\sqrt{2},0,7\pi/4),(\sqrt{2},0,0),(\sqrt{2},0,\pi/4)\}, \end{aligned} \tag{4}$$

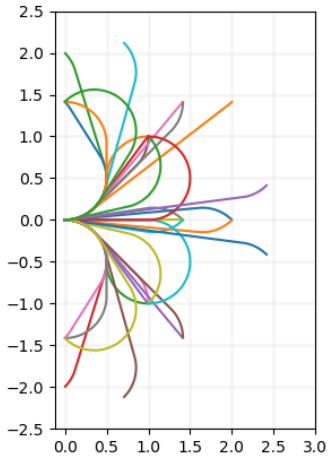
and a value of $t = 1.4$. The primitives obtained from the MILP consistently outperformed the standard primitives. We use a ratio of run times ($T_{\text{MILP}}/T_{\text{ROS}}$) of the A* algorithm as a metric for comparing the time efficiency of the two primitive sets, while a ratio of path lengths ($L_{\text{MILP}}/L_{\text{ROS}}$) provides a basis for comparison of performance. Of the 40 randomly generated maps, the average length and time ratios are

$$\left(\frac{L_{\text{MILP}}}{L_{\text{ROS}}}\right)_{\text{avg}} = 0.897, \text{ and } \left(\frac{T_{\text{MILP}}}{T_{\text{ROS}}}\right)_{\text{avg}} = 0.267.$$

The average path length using the MILP-obtained primitives was $\approx 90\%$ as long as those for the ROS package primitives and took $\approx 27\%$ of the time to calculate. In fact, in each of the 40 maps the MILP-obtained primitives consistently outperformed the ROS package primitives for both metrics. Figure 5 presents three example maps and paths. We notice that in addition to shorter faster solutions, the MILP-generated primitives also result in smoother paths with fewer turns. This fact will facilitate any further smoothing that is required.



(a) SBPL primitives.

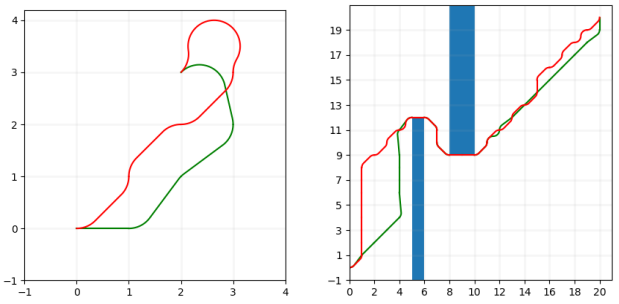


(b) MILP primitives.

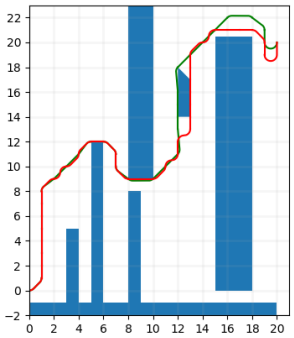
Fig. 4: (a): Motion primitives found in [17]. (b): Motion primitives determined from MILP (2)

C. Motion Primitives Euclidean Lattices

Another common problem is that of planning shortest paths in occupancy grids. The error between the shortest grid path and the optimal path depends on the number of neighbors considered for each gridpoint in the search. In [18, Figure 4], the authors present 4 and 8 neighbor grids in two dimensions, and 6 and 26 neighbor grids for three dimensions. They also provide error results in the form of t-values for these grid choices [18, Table 1]. For example, in the two dimensions, the 4 neighbor grid provides a t of ≈ 1.4, while the 8 neighbor grid provides a t of ≈ 1.08. Given that a $k \times k$ cubic grid in d dimensions can be modelled as a lattice of the form $L = \mathbb{Z}^d \cap [-k, k]^d$, generated by the canonical basis of $\mathbb{R}^d$, we can, for a given $t \geq 1$, determine a minimal set of $t-$spanning motion primitives using the MILP in (2). In Figure 6, we show the first quadrant of the neighbors needed to achieve t values of 1.0274, which results in a 16 neighbor grid, 1.0131, which results in a 24 neighbor grid, and 1.0124, which results in a 36 neighbor grid. These solutions extend the results in [18].



(a) Zero obstacles. (b) Two obstacles.



(c) Seven obstacles.

Fig. 5: Example paths for two sets of motion primitives. Red lines represent paths generated by standard motion primitives (see Figure 4a). Green lines represent paths generated using MILP-obtained motion primitives (see Figure 4b).

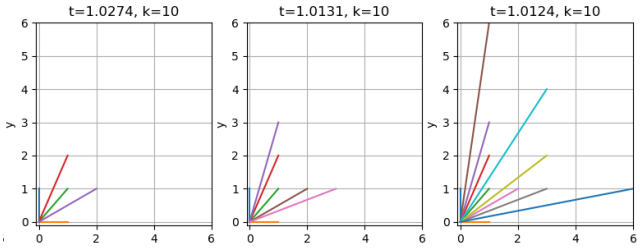


Fig. 6: Euclidean lattice minimal $t-$spanners in 2D and for different values of t.

VI. Conclusions and Future Work

The numerical examples presented in Section V illustrate the importance of minimal sets of $t-$spanning motion primitives in robotic motion planning. The MILP formulated in (2) represents the only known non-brute force approach to calculating *exact* minimal $t-$spanning motion primitives. Though solutions to (2) in general cannot be obtained in time polynomial in the size of the input lattice, motion primitives are generally calculated once, offline.

Observe that while minimal $t-$spanning motion primitives for a given lattice may be calculated using the MILP formulation in (2), the choice of lattice appears to be equally important to the time and length efficiency of path planning problems. The correct choice of lattice for a given mobile robot is a subject of future work.

There are lattices for which a solution to Problem II.5 may be efficiently obtained. Indeed, it can be shown that Problem II.5 for a Euclidean lattice with convex workspace is efficiently solvable regardless of the dimension. Observe that the proof of Theorem III.1, in which it is established that Problem III.1 is NP-hard, relies heavily on the potential non-convexity of the lattice workspace. The conditions under which Problem II.5 can be efficiently solved is still an open question.

References

[1] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[4] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3231–3237.

[5] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[6] M. Rufli and R. Siegwart, "On the design of deformable input-/state-lattice graphs," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3071–3077.

[7] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4889–4895.

[8] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 46–61, 2018.

[9] D. Peleg and A. A. Schäffer, "Graph spanners," *Journal of graph theory*, vol. 13, no. 1, pp. 99–116, 1989.

[10] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2172–2179.

[11] P. Carmi and L. Chaitman-Yerushalmi, "Minimum weight euclidean t-spanner is NP-hard," *Journal of Discrete Algorithms*, vol. 22, pp. 30–42, 2013.

[12] F. Bullo and A. D. Lewis, *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*. Springer Science & Business Media, 2004, vol. 49.

[13] L. Cai, "NP-completeness of minimum spanner problems," *Discrete Applied Mathematics*, vol. 48, no. 2, pp. 187–194, 1994.

[14] B. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial optimization*, 6th ed. Springer, 2018.

[15] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis, "Time constrained routing and scheduling," *Handbooks in operations research and management science*, vol. 8, pp. 35–139, 1995.

[16] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2019. [Online]. Available: http://www.gurobi.com

[17] Search-Based Planning Lab, "Search-based planning library (SBPL)," http://wiki.ros.org/sbpl, online, version 1.3.1 accessed 2019-02-28.

[18] A. Nash and S. Koenig, "Any-angle path planning," *AI Magazine*, vol. 34, no. 4, pp. 85–107, 2013.