Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

Locally Optimal Decomposition for Autonomous Obstacle Avoidance with the Tunnel-MILP Algorithm

Michael P. Vitus
Stanford University
Stanford, California, 94305

Steven L. Waslander
University of Waterloo
Waterloo, ON, Canada, N2L 3G1

Claire J. Tomlin
University of California
Berkeley, California, 94720

Abstract— **The Tunnel-MILP algorithm is a three stage path planning method for 2-D environments that relies on the identification of a sequence of convex polygons to form an obstacle free tunnel through which to plan a dynamically feasible path. This work investigates two aspects of the algorithm. First, a greedy cut method is proposed for improved decomposition of the environment, resulting in fewer regions than existing algorithms. Second, the effect of the decomposition on the resulting solution is investigated, and conditions are presented to demonstrate that the resulting tunnel cannot be improved to yield a better solution. This ensures that the tunnel provided does not adversely affect the resulting dynamically feasible trajectory, and guarantees local optimality of the solution.**

I. Introduction

With the increasing demand for autonomous vehicles, there is a pressing need for fast algorithms to plan trajectories through complex environments. Several examples of projects that require this technology are: search and rescue, reconnaissance, surveillance, and disaster response. These projects have motivated the Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC), a system used to test novel algorithms for multi-vehicle coordination and autonomous operation [1].

Obstacle avoidance is an essential technology for autonomous operation and one proposed solution to this problem applies mixed integer linear programming (MILP) ([2], [3]). It is well known that MILP is NP-hard in the number of binary variables required in the problem formulation [4], and so computational requirements grow exponentially as the number of binary variables needed to model the problem increases. Randomized methods, such as probabilistic roadmaps [5], have also been proposed to solve the obstacle avoidance problem, particularly for instances with a very large number of constraints. However, the resulting paths are generally not optimized in distance or time, and can result in winding paths if the vehicle dynamics are incorporated. To address these limitations, the Tunnel-MILP algorithm has been proposed. The algorithm constrains the MILP formulation by restricting the area in which the vehicle is allowed to travel, which significantly reduces the number of binary variables needed [6]. Using this restriction, a significant performance increase is realized over the standard MILP formulation. However, only a locally optimal solution can be guaranteed with the Tunnel-MILP algorithm.

Michael P. Vitus is a PhD. Candidate, Aeronautics and Astronautics. email: vitus@stanford.edu

Steven L. Waslander is an Assistant Professor in the Department of Mechanical and Mechatronics Engineering at the University of Waterloo. email: stevenw@uwaterloo.ca

Claire J. Tomlin is a Professor in Electrical Engineering and Computer Sciences. email: tomlin@eecs.berkeley.edu

The Tunnel-MILP algorithm is composed of the following three steps. First, a desirable preliminary path is planned through the environment ignoring the vehicle's dynamics. This pre-path can be computed by various methods; for example, a visibility graph approach [7], or gradient descent on a fast-marching potential [8]. Second, an obstacle free tunnel is formed around the pre-path as a sequence of convex polygons through which the vehicle must travel. Two examples of methods that obtain this convex decomposition are: trapezoidal decomposition [9] or constrained Delaunay triangulation [10]. Care must be taken in forming the decomposition as to not impose unnecessary restrictions on the final solution. Finally, a dynamically feasible trajectory is generated using a MILP formulation that restricts the vehicle to stay within the pre-defined tunnel.

This work presents two main contributions. First, a novel decomposition method is proposed that on average results in fewer polygons than trapezoidal decomposition or Delaunay triangulation. The proposed method is a greedy cut algorithm composed of three main steps: the decomposition is first restricted to the region surrounding the pre-path, cuts are then selected which guarantee division of the region into convex polygons, and the resulting polygons are then combined to form a tunnel around the pre-path. Second, conditions are presented under which the result of the Tunnel-MILP solution is shown to be locally optimal. Since the decomposition is formed prior to performing the optimization, there is potential for the resulting tunnel to affect the overall solution. The analysis is thus vital to ensure that the generated tunnel poses no unnecessary restrictions on the final solution, therefore guaranteeing a locally optimal solution for the chosen route through the obstacles.

The paper proceeds as follows. Section II presents the standard obstacle avoidance problem formulation. Then, each of the three stages of the Tunnel-MILP algorithm are described in detail in Section III. The greedy cut algorithm, a novel decomposition method, is presented in Section IV. In Section V, optimality conditions for the Tunnel-MILP solution are provided, and the paper concludes with directions of future work.

II. Problem Formulation

A. Environment

Consider a vehicle navigating through an environment bounded by a convex polygon, $E \subseteq \mathbb{R}^2$. An *edge* $e \subset \mathbb{R}^2$ is defined as a line segment with two end points, $e^1, e^2 \in \mathbb{R}^2$, that are not coincident. The boundary of an edge is $\delta e = \{e^1, e^2\}$ and its interior is $\text{int}(e) = e \setminus \delta e$. Let N^E be the number of edges of the environment polygon, $F^E \in \mathbb{R}^{N^E \times 2}$

be the outward pointing unit normals to each edge, and $G^E \in \mathbb{R}^{N^E}$ be the offset in the direction of the unit normals. Then the environment $E = \{x \in \mathbb{R}^2 | F^E x - G^E \leq 0\} \subseteq \mathbb{R}^2$ is defined by the intersection of halfspace constraints.

Similarly, obstacles can be defined that are entirely contained inside the environment, and must contain an interior point. Let N^O be the number of obstacles, and for each obstacle $O_i \in O$, let N_i^O be the number of edges. Each obstacle can then be defined analogously to the environment with components $F_i^O \in \mathbb{R}^{N_i^O \times 2}$ and $G_i^O \in \mathbb{R}^{N_i^O}$.

Finally, a planning space, P, for the vehicle can be defined as a polygon with holes. A *polygon with holes* is a connected, compact set in the plane whose boundary consists of an exterior polygonal contour and of finitely many interior polygonal contours, which are pairwise disjoint, noninclusive and nondegenerate. The planning space is then defined by a polygon with holes formed by removing the obstacles from the environment, $P = E \setminus \left\{ \bigcup_{i=1 \ldots N^O} O_i \right\}$. Let δP define the boundary of the set P. Note that the restriction to convex polygons for the environment and obstacles is included to ease the notation, although it is not necessary for the Tunnel-MILP algorithm or for the results of this work.

B. Vehicle Dynamics

The vehicle dynamics considered are motivated by the STARMAC platform [1], which can be approximated by standard linear point mass dynamics with a fixed discrete timestep of δt,

$$\begin{bmatrix} x(t+1) \\ v(t+1) \end{bmatrix} = A \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + Bu(t) \tag{1}$$

where $x(t) \in \mathbb{R}^2$ is the position, $v(t) \in \mathbb{R}^2$ is the velocity, $u(t) \in \mathbb{R}^2$ is the acceleration control. The position of the vehicle is constrained to lie in the planning space at each timestep, $t \in \{1, \ldots, T\}$, where T is the maximum number of timesteps allowed for the vehicle to reach the goal. The velocity and acceleration control inputs are also bounded. These constraints are summarized as,

$$\begin{aligned} & x(t) \in P\,, && \forall\, t \in \{1, \ldots, T\} \\ & \underline{v} \leq v_i(t) \leq \bar{v}\,, \quad && i \in \{1, 2\},\ \forall\, t \in \{1, \ldots, T\} \\ & \underline{u} \leq u_i(t) \leq \bar{u}\,, && i \in \{1, 2\},\ \forall\, t \in \{1, \ldots, T\}. \end{aligned} \tag{2}$$

C. General Problem

The general obstacle avoidance problem involves generating a trajectory for a vehicle operating under specified dynamics through a known obstacle-filled environment. The vehicle is assumed to start at some initial position and velocity and to have a known final goal, be it a final position or a final position and velocity, at unknown final time t_f. The objective function, which trades off minimum time and minimum control input, is as follows,

$$J(t_f, u) = \gamma t_f + (1 - \gamma) ||u||_1 \tag{3}$$

where $J : \mathbb{Z} \times \mathbb{R}^{2T} \to \mathbb{R}$, and $\gamma \in [0, 1]$. The general formulation of the path planning problem is,

General Program

$$\begin{aligned} \text{minimize} \quad & J(t_f, u) \\ \text{subject to} \quad & (1), (2) \end{aligned} \tag{P2.1}$$

III. Tunnel-MILP Algorithm

In order to remain outside convex obstacles, standard MILP formulations require the use of one binary variable per obstacle edge at every timestep. The main motivation behind the Tunnel-MILP algorithm is to reduce the number of binary variables needed by changing the constraints to require the state to remain inside an unobstructed set of convex polygons. Since the restriction of remaining inside a single convex polygon can be defined by a conjunction of linear constraints, the only binary variable needed is to indicate whether or not the vehicle is in the region. This formulation, therefore, requires only one binary variable per region for each timestep, resulting in reduced computational complexity compared with standard formulations. The Tunnel-MILP algorithm is composed of three steps as presented in Algorithm 1.

Algorithm 1 Tunnel-MILP Algorithm

```
1: Determine a pre-path through the environment ignoring
   the dynamics of the vehicle;
2: Decompose the environment into convex polygons
   around the pre-path;
3: Solve the dynamically feasible, optimal control problem
   through the sequence of convex polygons.
```

The first step of the Tunnel-MILP algorithm is to plan a pre-path through the planning space ignoring the vehicle's dynamics. There are various algorithms that can plan a not necessarily dynamically feasible path through the environment such as visibility graphs [7] and fast marching [8]. In both cases, the computation time is negligible compared with the final step of the Tunnel-MILP algorithm.

The second step is to decompose the environment into convex polygons and determine the sequence of polygons that entirely contains the pre-path, denoted as the tunnel $\tau = \{R_1, \ldots, R_\tau\}$. The regions $R_i\,, i \in \{1, \ldots, N^R\}$ denote the ordered sequence of polygons that comprise the tunnel, and each pair of consecutive regions must share a nondegenerate edge, $\text{int}(R_i \cap R_{i+1}) \neq \emptyset$. Let F_i^R and G_i^R define the edge constraints for each polygon, analogously to the region and environment constraints. There are various algorithms which accomplish this decomposition, two of which are trapezoidal decomposition and constrained Delaunay triangulation. An example of each of these decomposition methods is shown in Figure 1. Trapezoidal decomposition only uses cuts in one direction, typically vertical, to partition the environment, which usually leads to tall and narrow polygons to be included in the tunnel. Delaunay triangulation divides the environment into triangles, which are then greedily combined into convex polygons to form the tunnel. Section IV presents an alternative to these tunnel generation approaches that consistently returns a smaller number of polygons.

The final step is to solve the MILP optimization problem. The problem is formulated such that the vehicle is required to stay inside one region of the tunnel at each timestep, t, as follows,

$$\bigvee_{i \in \{1, \ldots, N^R\}} (F_i^R x(t) - G_i^R \leq 0)\,, \quad \forall t \in \{1, \ldots, T\} \tag{4}$$

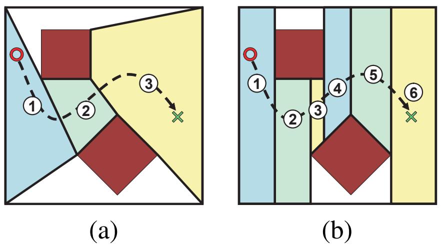

Fig. 1. An example of the tunnels generated from constrained Delaunay and trapezoidal decomposition. The obstacles (dark red), the pre-path (dashed), the sequence of polygons that form the tunnel (numbered) are visible. (a) Constrained Delaunay decomposition. (b) Trapezoidal decomposition.

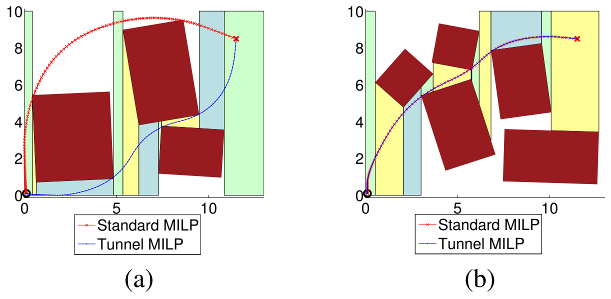

Fig. 2. Comparison between the path generated by the Tunnel-MILP algorithm and the globally optimal solution. The obstacles (dark red), tunnel regions are visible, along with the globally optimal trajectory (red x) and the Tunnel-MILP trajectory (blue-dotted).

Each OR-constraint is incorporated into the MILP formulation by introducing a binary variable [6]. Finally, to form the overall Tunnel-MILP optimization problem, Eqn. 4 needs to be added to the constraints of the general optimization problem P2.1.

Figure 2 compares the optimal path and the path generated by the Tunnel-MILP algorithm for three and five obstacle cases. In these examples, trapezoidal decomposition is used to form the tunnel. In Figure 2(a), the Tunnel-MILP's solution differs from the globally optimal solution because of the restriction imposed by the tunnel around the pre-path. In simulation results with various numbers of obstacles[1], the Tunnel-MILP algorithm often identified the globally optimal solution, and in the cases where it differed, it only increased the optimal time and input cost by an average of 3.3% and 3.8%, respectively. Computational cost savings of 59% were observed over a standard MILP formulation which yields the globally optimal solution upon completion.

IV. Tunnel Generation

The task of generating a tunnel through which to plan a dynamically feasible path can be seen as a decomposition of a polygon with holes into nonoverlapping, convex polygons in a way that results in the minimum number of regions along the pre-path. Similar problems have been posed in the field of computational geometry, and are often referred to as Optimal Convex Decomposition (OCD) problems [11]. Unfortunately, the decomposition of a general nonconvex polygon with non-rectilinear polygonal holes has been shown to be NP-hard [12]. Therefore, a greedy cut algorithm is developed which seeks to minimize the number of regions required to enclose the pre-path.

[1]All results were obtained from 100 random instances for each 3-6, 8, 9, and 20 obstacles.

The following definitions are needed. A *vertex* $\mu \in \mathbb{R}^2$ is defined by at least two edges which share a common end point. A nonconvex vertex is defined as one for which two of its edges form an interior angle greater that π without being bisected by another edge. A *cut* c is defined as a directed edge with start and end vertices, $\underline{\mu}_c$ and $\bar{\mu}_c$, that emanates from an existing vertex, $\underline{\mu}_c$, and ends at the first vertex, edge or preceding cut it encounters. If the end point of a cut is not coincident with an existing vertex, a new vertex is defined and the edge or cut that contains it is divided in two.

The pre-path defined by the first step of the Tunnel-MILP algorithm is defined as an ordered sequence of directed edges, $\rho = \{e_1, \ldots, e_{N_p}\}$, where N_p is the number of edges, and the end point of the edge e_i must be coincident with the start point of the edge e_{i+1}. Let $d_{max} = \sum_{j=1 \ldots N_p} ||e_j^2 - e_j^1||_2$ be the length of the pre-path. Then $d_\rho(x) : \mathbb{R}^2 \rightarrow [0, d_{max}]$ is defined as, $d_\rho(x) = \sum_{j=1 \ldots i-1} ||e_j^2 - e_j^1||_2 + ||x - e_i^1||_2$ for $x \in e_i$. Since d_ρ is a bijection, its inverse exists.

Definition 1: *The tunnel* τ ***encloses*** *the pre-path* ρ *if* $\forall x_1, x_2 \in \rho$ *given that* $x_1 \in R_i$, $x_2 \in R_j$, *where* $R_i, R_j \subset \tau$, *and* $d_\rho(x_2) > d_\rho(x_1)$ *such that* $d_\rho^{-1}(0) \in R_1$ *and* $j \geq i$. With this definition, it is possible for $R_i = R_j$ for $j \neq i$.

Previous work has shown that to decompose a polygon with holes into convex regions, it is sufficient to apply a single cut emanating from each nonconvex vertex in such a way that each of the two resulting interior angles is less than π [13]. The set of all cuts that eliminate a nonconvex vertex is denoted the *cone of bisection* and is depicted in Figure 3. It should be noted that each cut can eliminate at most two nonconvex vertices at a time.

A. Greedy Cut Algorithm

The greedy cut algorithm can now be defined in three main steps. First, the region of interest is restricted to contain only the area surrounding the pre-path. This step is optional, and is only used to reduce the problem complexity. Next, the set of nonconvex vertices is identified and cuts are selected to eliminate them while avoiding those that intersect the pre-path. Finally, once a sufficient number of cuts has been applied, the polygons that contain the pre-path are combined to form the tunnel in the MILP problem formulation.

1) Restriction: Once a pre-path is identified, a large portion of the planning space can often be eliminated. Having chosen a specific route through the planning space, much of the space that lies a significant distance from the pre-path is no longer relevant. This step of the algorithm generates a nonconvex polygon that encloses all area within a distance $l \in \mathbb{R}_+$ of the pre-path that is also inside the planning domain and outside of any obstacles. The results of this step of the algorithm are visible in Figure 3. The resulting nonconvex polygon, which may contain holes, is denoted the *feasible region* R_F.

2) Cut Selection: Two types of cuts are considered. A cut that eliminates two nonconvex vertices simultaneously is

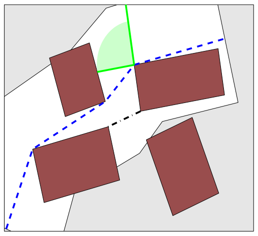

Fig. 3. The feasible region (white) for a specific example with four obstacles (dark red) and a pre-path (dashed blue). A matching cut (dash-dotted line), two extreme cuts (light solid green) and the cone of bisection (light green) are also displayed.

referred to as a *matching cut*, and can exist if and only if the edge connecting both vertices lies within the cone of bisection for each vertex. The aim of this set of cuts is to reduce the total number of cuts applied to the feasible domain, and thereby reducing the total number of convex polygons. The second type of cut considered is denoted an *extreme cut*, and is defined as either boundary of the cone of bisection. With this set of cuts, it is possible to continue adding cuts until no nonconvex vertices remain, and thereby complete the decomposition of the feasible region into convex polygons.

In determining which cuts to select for each non-convex vertex, Lemma 1 provides useful insight. Let B_x^ϵ denote the ball of radius ϵ about a point x.

Definition 2 (Edge Crossing): *Let e be an edge for which $A_e x = b_e, \forall x \in e$ and let p be a path. Then, $y \in int(e)$ is an* ***edge crossing point*** *of p if $\forall \epsilon > 0, \exists y', y'' \in B_y^\epsilon \cap p$ such that $A_e y' - b_e < 0$ and $A_e y'' - b_e > 0$.*

Lemma 1: *Given a convex decomposition of the feasible region, R_F, the number of convex regions that contain the pre-path is bounded below by the number of edges that contain an edge crossing point of the pre-path ρ plus one.*

Proof: Let τ be the tunnel that contains the pre-path ρ, and let $y \in \rho$ be an edge crossing point on the edge e. Since y is an interior point of $e \subset R_F$, there exists an ϵ such that $B_y^\epsilon \subset R_F$. By the definition of an edge crossing point, $\exists y', y'' \in B_y^\epsilon \cap \rho$ such that $A_e y' - b_e < 0$, and $A_e y'' - b_e > 0$. Both y' and y'' are in R_F, and therefore a region R_i exists for which $y' \in R_i$. By compactness of the region R_i, the edge must be on the boundary of R_i, that is $e \subset \delta R_i$. But $y'' \notin R_i$ because $A_e y'' - b_e > 0$. This implies $y'' \in R_j$, and the region R_j must be included in the tunnel enclosing ρ. Since the starting point of ρ must lie in a convex region of R_F and each subsequent edge crossing along the path must add a region to τ, the number of regions needed to enclose the path ρ is bounded below by the number of edge crossings plus one. ∎

As demonstrated by Lemma 1, each cut that contains an edge crossing of the pre-path adds an additional region to the tunnel containing it. Therefore, avoiding cuts that intersect the pre-path ensures the greedy algorithm does not unnecessarily increase the total number of polygons in the tunnel, thereby increasing the size of the MILP formulation. Although this Lemma does not cover all cases of cuts that intersect the path, the remaining cases are omitted for brevity.

The cuts are applied in the following greedy manner. The set of nonconvex vertices are ordered based on distance to the pre-path, breaking ties in order from start to finish along the pre-path. All matching cuts are identified at each vertex, and any that do not intersect the pre-path are applied. A random selection is made if more than one matching cut is found. A cut c is applied by including it in the set of edges that defines the feasible region, R_F, so that it too may stop the propagation of subsequent cuts. If a matching cut does intersect the pre-path, then the extreme cuts for both vertices are tested to see if one can be found for each vertex that does not intersect the pre-path. If not, the matching cut is applied. Once the matching cuts have been applied, extreme cuts are applied at each remaining nonconvex vertex in the same vertex order, selecting cuts that do not cross the pre-path whenever available.

3) Polygon Identification: Finally, the sequence of convex polygons that completely contains the pre-path are identified. This is achieved by identifying all convex polygons in the decomposition, and searching along the pre-path to identify all polygons through which the pre-path passes.

B. Results Comparison

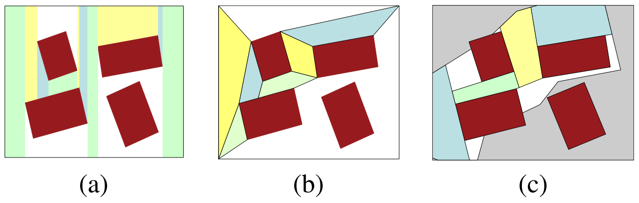

Fig. 4. The sequence of tunnels generated by (a) trapezoidal, (b) constrained Delaunay and (c) greedy-cut decomposition. In total, 10 regions are created by trapezoidal decomposition, 6 are created by constrained Delaunay decomposition, and 4 are created by the greedy-cut method.

A comparison of the greedy cut algorithm with two alternative methods of tunnel generation is presented in Figure 4. For the four obstacle example chosen, it is clear that the greedy cut algorithm significantly reduces the number of polygons that comprise the tunnel, from 10 for trapezoidal decomposition and 6 for constrained Delaunay triangulation to 4. It is interesting to note that many of the cuts generated by the other two methods do not fall in the cone of bisection for the corresponding nonconvex vertex, resulting in a larger number of cuts needed to ensure convexity of the polygonal decomposition. Table I presents the average number of polygons required to enclose the pre-path for 50 test cases with four and eight obstacles, using the greedy cut, constrained Delaunay and trapezoidal decompositions. The greedy cut algorithm demonstrates a 31% and 58% improvement over constrained Delaunay and trapezoidal methods for four obstacle environments. For eight obstacle environments, the improvements were 20% and 62%, respectively. In each case, the number of binary decision variables in the final stage of the Tunnel-MILP algorithm were reduced proportionately.

V. Optimality of Tunnel-MILP Solution

The Tunnel-MILP algorithm can at best only provide a locally optimal solution, which is dependent on the arbitrary constraints imposed by the decomposition of the space

TABLE I
AVERAGE NUMBER OF TUNNEL REGIONS FOR THE GREEDY CUT, CONSTRAINED DELAUNAY AND TRAPEZOIDAL DECOMPOSITIONS.

	4 Obstacles	8 Obstacles
Greedy Cut	4.4	6.0
Delaunay	6.3	7.5
Trapezoidal	10.5	15.9

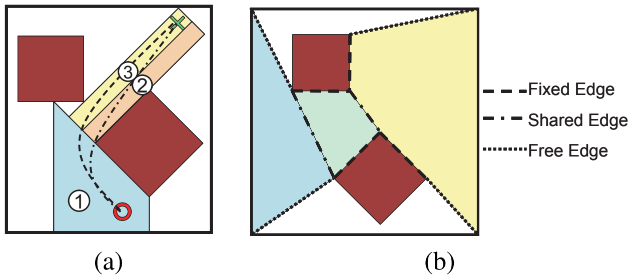

Fig. 5. a) Different decomposition for the same environment which leads to two different locally optimal paths. b) Classification of three different edge types.

around the pre-path. Figure 5(a) shows an example of two different decompositions, composed of the polygons $(1, 3)$ or $(1, 2, 3)$, which yield two different solutions. Therefore, it is important to understand the impact of the tunnel constraints on the optimality of the solution.

In the following section, conditions are provided that ensure the local optimality of the solution for any decomposition around the pre-path. In other words, the solution cannot be improved by changing the arbitrary constraints imposed by the decomposition. First, the underlying subproblem for the Tunnel-MILP optimization is defined, and convex sensitivity analysis is presented. Next a formal definition of the three different edge constraints (fixed, shared and free), shown in Figure 5(b), are defined. Then, it will be shown that only the active, free edge constraints can decrease the value of the objective function. Therefore, if there are no active, free edge constraints, then the trajectory cannot be improved by modifying the tunnel.

A. Sub-Problem

If all of the decision variables are assumed to be known, i.e. which region the vehicle will be in at each time step and the optimal final time, then the problem simplifies to a convex optimization problem. It will also be assumed that the subproblem is feasible since the decision variables are known. Let $\tilde{F}_t^R$ and $\tilde{G}_t^R$ be the region constraints for the polygon that the vehicle is in at timestep t. Under these assumptions, the Tunnel-MILP Program simplifies to

Convex Subproblem

$$\begin{array}{lc}\text{minimize} & ||u||_1 \\ \text{subject to} & (1),\ (2) \\ & \tilde{F}_t^R x(t) - \tilde{G}_t^R \leq 0 \quad \forall t \in \{1, \ldots, t_f\}\end{array} \qquad \text{(P5.1)}$$

The edges of a region within the polygonal tunnel shall be denoted e_{ij}, and the region that the vehicle is in at timestep t is denoted R_t.

B. Sensitivity Analysis

Sensitivity analysis is useful for determining which constraints, if perturbed, will impact the value of the objective function. In order to use sensitivity analysis, strong duality needs to be shown for the convex subproblem (P5.1). Since the subproblem is feasible and can be transformed into a linear program, it can be shown by using the weaker form of Slater's condition that strong duality holds [14].

To apply sensitivity analysis, the polygon inequality constraints will be perturbed, $\tilde{F}_t^R x(t) - \tilde{G}_t^R \leq w_t$, and the optimal value of the perturbed objective function is represented by $\tilde{J}^*(w)$. Assuming the dual optimum is feasible and λ^* is optimal for the dual of the unperturbed problem, then for all w there exists a bound on the perturbed objective function:

$$\tilde{J}^*(w) \geq \tilde{J}^*(0) - \sum_{t=0}^{T} \lambda_t^{*T} w_t \qquad (5)$$

By definition, an inequality constraint is considered *active* if it holds with equality and *inactive* otherwise. Note that Lagrange multipliers are non-negative and are equal to zero for inactive constraints.

C. Constraint Classification

The perturbations, w_t, are of the edges of the tunnel regions. There are three different types of edges: fixed, shared and free. An example of each type of edge is shown in Figure 5(b). All edges of the polygons that compose the tunnel can be classified as one of the three edge types.

Definition 3 (Fixed Edge): *The edge, e_{ij}, is a **fixed edge** of a tunnel polygon if it coincides with an obstacle edge or a boundary edge for its entire length. That is, $e_{ij} \in \delta P$.*

Definition 4 (Shared Edge): *The edge, e_{ij}, is a **shared edge** of a tunnel polygon if it coincides with another tunnel polygon edge for its entire length. That is for the edge, $\exists k \in \{1, \ldots, N^R\}, m \in \{1, \ldots, N_k^R\}$ where $k \neq i$ such that: $e_{ij} = e_{km}$.*

Definition 5 (Free Edge): *The edge, e_{ij}, is a **free edge** of a tunnel polygon if it does not coincide with an obstacle or boundary edge and is not a subset of any other region. That is, $e_{ij} \notin (\delta P \bigcup_{k \neq i,\, k \in \{1,\ldots,N^R\}} \delta R_k)$.*

D. Local Optimality of Tunnel Solution

The following four lemmas will determine which constraints can decrease the value of the objective function.

Lemma 2: *An inactive edge, e_{tj}, cannot be perturbed in a direction that will decrease the value of the objective function, $\tilde{J}^*(w) \geq \tilde{J}^*(0)$.*

Proof: Assume without loss of generality that all other edges are not perturbed. By definition, an inactive edge's Lagrange multiplier is zero, $\lambda_{tj}^* = 0$. From Equation (5), $\tilde{J}^*(w) \geq \tilde{J}^*(0) - \lambda_{tj}^* w_{tj} \geq \tilde{J}^*(0)$. Therefore, a perturbation of an *inactive* region constraint may increase the value of the objective function, but never decrease it. ■

Definition 6 (Feasible Perturbation): *Let C be the complement of the planning space defined by $C = \mathbb{R}^2 \setminus P$, R be a polygon defined by $R = \{x \in \mathbb{R}^2 | F_i^R x - G_i^R \leq 0\}$ and $\tilde{R}$ be the corresponding polygon after the perturbation, w_i, is applied be defined by $\tilde{R} = \{x \in \mathbb{R}^2 | F_i^R x - G_i^R \leq w_i\}$. A **feasible perturbation**, w_i, is one for which, $\tilde{R} \cap C = \emptyset$.*

Lemma 3: *For every active, fixed edge, e_{tj}, no feasible perturbation, w_{tj}, exists that decreases the value of the objective function, $\tilde{J}^*(w) \geq \tilde{J}^*(0)$.*

Proof: Assume without loss of generality that only the active, fixed edge, e_{tj} is perturbed. From Equation (5), a bound on the optimal value of the objective function for this perturbation is: $\tilde{J}^*(w) \geq \tilde{J}^*(0) - \lambda^*_{tj} w_{tj}$. The outward pointing normal of the edge, $\tilde{F}^R_{tj}$, points into the restricted area, C. Therefore, the only feasible perturbation for the edge is $w_{tj} < 0$. Since $\lambda^*_{tj} \geq 0$, it follows that, $\tilde{J}^*(w) \geq \tilde{J}^*(0)$. Therefore, a feasible perturbation of a fixed edge will never decrease the value of the objective function. ■

Lemma 4: *Adding a shared edge, e_{ij}, to the tunnel does not impact the value of the objective function.*

Proof: Let the feasible region for the location of the vehicle be defined as, $\bar{R} = \bigcup_{i=1 \ldots N^R} R_i$. Consider the Tunnel-MILP optimization problem, in which the vehicle is restricted to be in the tunnel $\bar{R}$, $\forall\, t \in \{1, \ldots, T\}$. Let $x^*(t)$ be the optimal position of the vehicle at time t, and let the optimal value of the objective function be J^*.

Now consider a modification of the problem in which at timestep t_1 the vehicle is restricted to be on a shared edge, e_{ij}. Assume that $x^*(t_1) \in e_{ij}$. Since the feasible region has been restricted in the modified problem, $J^* \leq \hat{J}^*$ where $\hat{J}^*$ is the optimal value of the objective function for the modified problem. However, x^* is still feasible in the modified problem, consequently: $J^* = \hat{J}^*$. Therefore, adding shared edges to the tunnel, $\bar{R}$, does not affect the value of the objective function even if they are active. ■

Lemma 5: *An active, free edge, e_{tj}, can be perturbed in a direction that may decrease the value of the objective function, $\tilde{J}^*(w) \geq \tilde{J}^*(0) - \Delta$ where $\Delta \geq 0$.*

Proof: Assume without loss of generality that only the active, free edge, e_{tj} is perturbed. The feasible perturbations for the edge are: $w_{tj} > 0$ and $w_{tj} < 0$. From Equation (5), a bound on the optimal value of the objective function for this perturbation is: $\tilde{J}^*(w) \geq \tilde{J}^*(0) - \lambda^*_{tj} w_{tj}$. For $w_{tj} > 0$ and since $\lambda^*_{tj} \geq 0$ by definition, $\lambda^*_{tj} w_{tj} \geq 0$. Therefore, there exists a perturbation of an active, free edge that may decrease the value of the objective function. ■
An intuitive explanation of the result from Lemma 5 is that a perturbation of $w_{tj} > 0$ increases the size of the polygon region, which provides more space for the vehicle to find a better path to the goal.

Theorem 1: *If there are no active free edge constraints, then there are no perturbations that exist such that $\tilde{J}^*(w) < \tilde{J}^*(0)$. Hence, the locally optimal solution for any decomposition around the pre-path has been achieved.*

Proof: From Lemmas 2, 3, 4 and 5, it was shown that only the set of active, free edges can affect the solution. Therefore, if there are no active, free edge constraints, then the solution cannot be improved to reduce the value of the objective function, that is, $\tilde{J}^*(w) \geq \tilde{J}^*(0)$. ■

Corollary 1: *If the globally optimal path is through the polygon decomposition around the pre-path, then the solution obtained from the Tunnel-MILP algorithm will be the globally optimal solution.*

Using the optimality conditions derived, an algorithm can be devised which is guaranteed to produce the locally optimal solution for all decompositions around the pre-path for any initial decomposition.

VI. Conclusions

The Tunnel-MILP algorithm, which approximates the obstacle avoidance problem, is dependent on the decomposition of the space around the pre-path. A suboptimal decomposition method was proposed, and optimality conditions were developed to show that only the active, free edge constraints can decrease the value of the objective function.

An area of interest for future work is to determine a maximum bound on the minimum number of convex polygons needed to decompose the space around the pre-path, which can be used to formulate the maximum number of binary variables needed. Finally, for applications such as those envisaged for the STARMAC testbed, extension to three dimensions is vital, and will require alternative methodologies for pre-path generation algorithm as well as for the convex decomposition of the planning space.

Acknowledgements

Special thanks to Kaushik Roy for numerous discussions.

References

[1] G. Hoffmann, H. Huang, S. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, (Hilton Head, South Carolina), August 2007.

[2] A. Richards, Y. Kuwata, and J. How, "Experimental demonstrations of real-time MILP control," in *Proceeding of the AIAA Guidance, Navigation, and Control Conference*, August 2003.

[3] M. G. Earl and R. D'Andrea, "Iterative MILP methods for vehicle-control problems," *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1158–1167, 2005.

[4] M. R. Garey and D. S. Johnson, *Computers And Intractability: A guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman and Co., 1979.

[5] L. E. Kavraki, P. Svestka, J.-C. 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.

[6] M. P. Vitus, V. Pradeep, G. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Tunnel-MILP: Path planning with sequential convex polytopes," in *In the Proceedings of the AIAA Guidance, Navigation, and Control Conference*, (Honolulu, Hawaii, USA), 2008.

[7] T. Asand, T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility of disjoint polygons," *Algorithmica*, vol. 1, pp. 49–63, 1986.

[8] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. New York, NY, USA: Springer-Verlag, 2002.

[9] B. Chazelle, *Advances in Robotics, Vol.1: Algorithmic and Geometric Aspects of Robotics*, ch. Approximation and Decomposition of Shapes, pp. 145–185. Lawrence Erlbaum Associates, 1987.

[10] L. P. Chew, "Constrained delaunay triangulations," in *Proceedings of the Third Annual Symposium on Computational geometry*, (New York, NY, USA), pp. 215–222, ACM, 1987.

[11] B. Chazelle and D. P. Dobkin, "Optimal convex decompositions," in *Computatinoal Geometry* (G. Toussaint, ed.), pp. 63–133, Amsterdam: North-Holland, 1985.

[12] A. Lingas, "The power of non-rectilinear holes," *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, vol. 140, pp. 369–383, 1982.

[13] H. Martini and V. Soltan, "Minimum convex partition of polygonal domains by guillotine cuts," *Discrete and Computational Geometry*, vol. 19, no. 2, pp. 291–305, 1998.

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, England: Cambridge University Press, 2004.