Incremental Path Repair in Hierarchical Motion-Planning with Dynamical Feasibility Guarantees for Mobile Robotic Vehicles

Zetian Zhang* and Raghvendra V. Cowlagi*

***Abstract*— New requirements of autonomous mobile vehicles necessitate hierarchical motion-planning techniques that not only find a plan to satisfy high-level specifications, but also guarantee that this plan is suitable for execution under vehicle dynamical constraints. In this context, the H-cost motion-planning technique has been reported in the recent literature. We propose an *incremental* motion-planning algorithm based on this technique. The proposed algorithm retains the benefits of the original technique, and also significantly reduces the associated computational cost. In particular, the proposed iterative algorithm presents during intermediate iterations *feasible* motion-planning solutions, with the guarantee that the algorithm eventually converges to an optimal solution. Furthermore, the costs of solutions at intermediate iterations are nonincreasing except, possibly, at a finite number of special iterations. Therefore, the proposed algorithm is suitable for real-time implementations, where hard bounds on the available computation time may be imposed, and where the original H-cost optimization algorithm may not have sufficient time to converge to any solution at all. We illustrate the proposed algorithm with numerical simulation examples.**

I. Introduction

The capability of autonomous motion envisioned for mobile robotic vehicles has evolved from the traditional idea of point-to-point motion (e.g. *"go from A to B"*) to the more general idea of a completing a *task* [1]. Such a task (e.g. *"monitor region A until events B or C occur, then go to regions D or E ..."*) is typically specified using formulae of predicate- or temporal logic.

In this context, the scope of *motion-planning problem* for autonomous mobile vehicles has expanded beyond that of the traditional constrained trajectory optimization problem. Specifically, the motion-planning problem consists of two distinct parts: (1) the *task-planning problem* of finding a finite sequence of actions – namely, a *plan* – to complete the assigned task, and (2) the *trajectory generation problem* of finding a vehicle trajectory to execute this plan. The task-planning problem is formulated as a combinatorial optimization problem, e.g. path optimization in a graph. The trajectory generation problem is formulated as a problem of finding optimal control inputs for a dynamical system Γ that models the vehicle's motion.

The task-planning and trajectory generation parts of the motion-planning problem are formulated on fundamentally different domains, and it is therefore beneficial to hierarchically separate the solutions of these two parts, cf. [2]–[9]. In such hierarchically separated solutions, task-planning is achieved, for example, using geometric methods such as workspace cell decomposition [10]. Briefly, workspace cell decomposition involves partitioning of the vehicle's 2D or 3D workspace into convex regions called *cells*. Each cell is either completely filled with obstacles or freely traversable by the vehicle. A graph is constructed by associated each traversable cell with a unique vertex of this graph, and by defining edges based on geometric adjacency of cells.

Hierarchical separation in motion-planning entails the need to maintain "compatibility" between the task-planning and trajectory generation algorithms. To this end, the H-cost motion-planning technique has been recently reported [11]. This technique is based on workspace cell decompositions, and the central idea is to incorporate characteristics of the vehicle's physical motion capabilities into the high-level planner via transition costs assigned to successions of edges in the graph associated with workspace cell decomposition. The H-cost technique involves path optimization on a so-called *lifted graph* (see Section II). The vehicle models considered in [11] include the Dubins car model, a particle dynamical model with acceleration constraints, and an aircraft dynamical model.

The H-cost motion-planning technique closely aligns with the recently developed idea of using discrete abstractions [12], [13] of vehicle dynamical system models in hierarchical motion-planning [14]–[18]. Informally, an abstraction results in a finite state transition system $\mathcal{G}$, which represents the evolution of the dynamical system Γ that models the vehicle's physical motion. In [14], [15], this state transition system is constructed as follows: the workspace is partitioned, and a graph is constructed to represent the motion of the state of Γ over the subregions of this partition via admissible state trajectories. In [16]–[18], a similar approach is followed, except that the *state space* is partitioned. The high-level task is specified by an LTL formula ϕ, and another state-transition system, so-called a Büchi automaton, is constructed. Informally, the Büchi automaton admits only those transitions that satisfy ϕ. Finally, the state transition system $\mathcal{G}$ for the overall motion-planning problem is constructed as the product of the aforesaid graph (based on partitioning) and the Büchi automaton. Consequently, any high-level plan found by searching $\mathcal{G}$ is guaranteed to be "compatible" with the vehicle dynamical system model, in that there are guaranteed (by construction) to exist vehicle trajectories that can execute the high-level plan. In the context of the H-cost technique, future motion-planning algorithms involving product transition systems of a Büchi automaton with the lifted graph can be envisioned.

Computational inefficiency is a serious concern in current

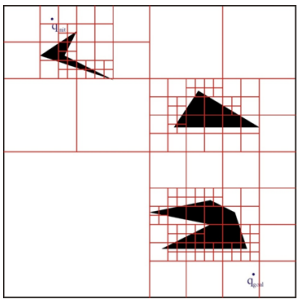
*Aerospace Engineering Program, Worcester Polytechnic Institute, Worcester, MA, USA. {zzhang, rvcowlagi}@wpi.edu

hierarchical motion-planning approaches that make explicit guarantees of "compatibility" between the task-planning and trajectory generation algorithms. For example, the size of the product automaton in [16]–[18] increases combinatorially with the LTL alphabet involved in ϕ. In the H-cost technique, the size of the lifted graph increases exponentially with a main parameter of the technique. Improvements in the computational efficiency of the abstraction-based approach [16]–[18] have been recently achieved via receding horizon ideas [19], [20]. Nevertheless, these methods are restrictive in the sense that they are based on the construction of control laws that do not include notions of trajectory costs for Γ. Also, abstractions as defined in [14], [16] *may not exist* for vehicles with nonholonomic constraints.

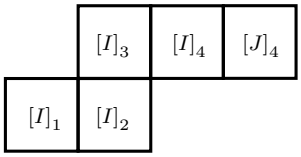
In this paper, we propose the idea of *incremental* H-cost motion-planning. The proposed algorithm retains the primary benefit of the original H-cost technique [11] of finding an optimal high-level plan with the guarantee that there exists a state trajectory of Γ to execute this plan. Additionally, the proposed algorithm significantly mitigates the computational cost of the original H-cost approach by introducing incremental computations. The proposed algorithm is incremental in that it produces a sequence of *feasible* plans in intermediate iterations, which asymptotically converge to an *optimal* plan. Therefore, the proposed algorithm is *eventually optimal*, and it is also suitable for real-time implementation with a hard bound on the available computation time. The primary motivation behind developing an incremental algorithm is as follows: the fundamental computational complexity in the H-cost technique (or any similar hierarchical motion-planning technique) cannot be defeated if the objective is solely to find an *optimal* motion plan. In the interests of real-time implementation, it is beneficial to develop a motion planner that returns a feasible solution at nearly any time during its search for an optimal motion plan.

The proposed algorithm relies on iterative repair of paths in lifted graphs. We focus on the point-to-point motion-planning problem for developing the algorithm; as previously stated, it is easy to envision extensions to the general problem where the high-level task is specified by logic formulae. The H-cost algorithm [11] attempts to find optimal paths in lifted graphs with appropriately defined edge costs. Whereas the search for such optimal paths is inherently difficult, the proposed approach alleviates this difficulty by iteratively "seeding" the search with the result of a simpler optimization problem, and then replacing high-cost edges in the "seed" path. This idea of path repair has been discussed in different contexts in [21], [22].

The contributions of this paper are as follows. Firstly, we propose a novel, computationally efficient, and hierarchically separated solution to the point-to-point motion-planning problem. The proposed approach can incorporate complex vehicle dynamical characteristics, and it guarantees "compatibility" between the ask-planning and trajectory generation algorithms. Furthermore, the proposed approach possesses the desirable property of maintaining feasible solutions in intermediate iterations while asymptotically converging to an optimal solution. Secondly, we elucidate the properties of lifted graphs as a general tool for hierarchical motion-planning involving different task-planning problems and vehicle dynamics. Finally, we discuss the suitability of the proposed approach to future real-time computations with bounds on the computation time. Specifically, the proposed approach can be implemented to use as much computational time as available, instead of requiring a significant minimum period for arriving at a usable result.



(a) Cell decomposition with square cells.



(b) A sequence of cells associated with $(I, J) \in E_3$. Here $[I]_2 = [J]_1$, $[I]_3 = [J]_2$, and $[I]_4 = [J]_3$.

Fig. 1. Cell decomposition and the geometric association of H-histories.

The rest of this paper is organized as follows. In Section II, we briefly review the H-cost motion-planning algorithm, including the idea of lifted graphs. In Section III, we present the proposed incremental H-cost algorithm and the main result of the paper. In Section IV, we present examples that illustrate the benefits of the proposed algorithm. We conclude the paper in Section V with comments on future work.

II. Problem Formulation

The H-cost motion-planning technique is based on workspace cell decomposition, i.e. a partition of the workspace into convex subregions called *cells*, such that each cell is either full of obstacles or freely traversable. To precisely state the motion-planning in this context, let $\mathcal{W}$ denote a planar region (namely, the workspace) with obstacles in which the vehicle moves, and consider a cell decomposition of this region. Following the notation and terminology in [11], we associate with this partition a graph $\mathcal{G} := (V, E)$ such that each vertex of $\mathcal{G}$ is uniquely associated with an obstacle-free cell, and each edge of $\mathcal{G}$ is uniquely associated with a pair of geometrically adjacent cells. We denote by $\mathsf{cell}(v)$ the subregion of $\mathcal{W}$ associated with the vertex $v \in V$.

A *path* π in $\mathcal{G}$ is a finite sequence $(j_0, j_1, \ldots, j_P)$ of vertices with no repetition such that $j_0, j_k \in V$, and $(j_{k-1}, j_k) \in E$, for each $k = 1, \ldots, P$. In general, a task-planning problem can be formulated as the problem of finding a path in $\mathcal{G}$ that satisfies certain criteria. In particular, the point-to-point motion-planning problem, at the higher level of planning, is the problem of finding a path in the graph $\mathcal{G}$ with minimum cost and with prespecified initial and final vertices. We denote by $\mathcal{L}^*$ the set of all paths in the graph $\mathcal{G}$, and by $\mathcal{L}^*(j_0, j_P)$ the set of all paths with initial

vertex $j_0 \in V$ and final vertex $j_P \in V$. The *cost* $\mathcal{J}_0 : \mathcal{L}^* \to \mathbb{R}_+$ of a path is defined by $\mathcal{J}_0(\pi) := \sum_{k=1}^{P} g_0(j_{k-1}, j_k)$, where $g_0 : E \to \mathbb{R}_+$ is a prespecified edge cost function. We refer to this problem, which has been well studied in the literature (for instance, [23]), as the *standard optimal path problem*. The problem can be solved, for instance, by Dijkstra's algorithm.

To introduce H-costs, for every integer $H \geqslant 0$, let

$$V_H := \{(j_0, \ldots, j_H) : (j_{k-1}, j_k) \in E,\ k = 1, \ldots, H, \\ j_k \neq j_m, \quad \text{for } k, m \in \{0, \ldots, H\}, \text{ with } k \neq m\}.$$

Every element I of the set V_H is an ordered $(H+1)$-tuple of the elements of V, and this tuple corresponds to a sequence of cells (see Fig. 1). Following [11], we denote by $[I]_k$ the k^{th} element of I, and by $[I]_k^m$ the tuple $\big([I]_k, [I]_{k+1}, \ldots, [I]_m\big)$, for $k < m \leqslant H+1$. Let E_H be a set of all pairs (I, J), with $I, J \in V_H$, such that $[I]_k = [J]_{k-1}$, for every $k = 2, \ldots, H+1$, and $[I]_1 \neq [J]_{H+1}$. An element of the set E_H is called an *H-history*. Note that each H-history is an ordered $(H+2)$-tuple of elements of V, i.e., a succession of $(H+1)$ edges in the graph $\mathcal{G}$. According to this notation, $V_0 = V$ and $E_0 = E$. Also note that the largest integer $\bar{H}$ for which these definitions remain meaningful is the diameter of the graph $\mathcal{G}$, i.e. $\bar{H} = \operatorname{diam}(\mathcal{G})$. The diameter of a graph is the largest distance (in terms of number of edges in the path with the least number of edges) between any two vertices in the graph.

Let $g_H : E_H \to \mathbb{R}_+$ be a non-negative function.

Definition 1. The *H-cost* of a path π in $\mathcal{G}$ is defined by

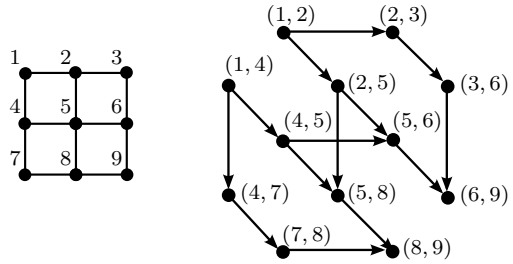
$$\mathcal{J}_H(\pi) := H + \sum_{k=H+1}^{P} g_H\left(j_{k-H-1}, j_{k-H}, \ldots, j_k\right). \quad (1)$$

For every $H \geqslant P$, we define $\mathcal{J}_H(\pi) = \mathcal{J}_{P-1}(\pi)$. In [11], the point-to-point motion-planning problem is formulated as that of finding a path in $\mathcal{G}$ with minimum H-cost and with prespecified initial and final vertices. We refer to this problem as the *H-cost optimal path problem*. Suitable definitions of g_H are crucial for the application this optimization problem in motion-planning, which we will discuss later.

A. Vehicle Model

As in [11], let $(x, y, \theta) \in \mathcal{C} := \mathbb{R}^2 \times \mathbb{S}^1$ denote the configuration (i.e. the position of the vehicle center of mass and the direction of its velocity vector) of the vehicle in a prespecified Cartesian coordinate system, and let ψ denote any additional state variables required to describe the state of the vehicle. The vehicle model is a controlled nonlinear dynamical system, which we denote by Γ. We assume that $\psi \in \Psi$, where Ψ is a n-dimensional smooth manifold. The state of the system Γ is $\xi := (x, y, \theta, \psi) \in \mathcal{D} := \mathcal{C} \times \Psi$.

Let $U \in \mathbb{R}^m$ denote the set of admissible control values, and for $t > 0$, let $\mathcal{U}_{[t_1,t_2]}$ denote the set of piecewise continuous functions defined on the interval $[t_1, t_2]$ that take values in U. The evolution of the system Γ is described by the differential equation $\dot{\xi}(t) = f(\xi(t), u(t))$ for all $t \geqslant [0, t_\text{f}]$, where $u \in \mathcal{U}_{[0,t_\text{f}]}$ is an admissible control input, and f is sufficiently smooth to guarantee global existence and uniqueness of solutions. We denote by $\xi(\cdot\,; \xi_0, u)$ the unique state trajectory with initial condition $\xi(0) = \xi_0$, and by $\mathsf{x}(\xi)$ the projection on $\mathbb{R}^2$ of a state $\xi \in \mathcal{D}$.



(a) Original graph $\mathcal{G}$. (b) Lifted graph $\mathcal{G}_1$.

Fig. 2. Example of a lifted graph (all vertices of $\mathcal{G}_1$ are not shown).

B. Lifted Graph

The *lifted graph* $\mathcal{G}_H$, as defined in [11], is a directed graph whose vertex and edge sets are, respectively, V_H and E_H (Fig. 2 shows an example of a lifted graph with $H = 1$). Note that the definition of the lifted graph $\mathcal{G}_H$ is meaningful only with reference to the original graph $\mathcal{G}$, and also that $\mathcal{G}_0 = \mathcal{G}$.

Every path $\pi^0 = (j_0, \ldots, j_P)$ in the original graph $\mathcal{G}_0$ can be uniquely mapped to a path $\pi^H = (J_H, \ldots, J_P)$ in the lifted graph $\mathcal{G}_H$, where $J_k := (j_{k-H}, \ldots, j_k) \in V_H$ for each $k = H, \ldots, P$. We denote this map by $b_0^H : \mathcal{L}_0^* \to \mathcal{L}_H^*$, where $\mathcal{L}_H^*$ is the set of all paths in $\mathcal{G}_H$. Therefore, $\pi^H = b_0^H(\pi^0)$ and $\pi^0 = b_H^0(\pi^H)$. For every integer $H \geqslant 0$, the map b_0^H is bijective and b_H^H is the identity map. For integers $H, L \geqslant 0$, we define $b_H^L := b_0^L \circ b_H^0$. In general, the following property is true for integers $H, L, M \geqslant 0$: $b_L^M \circ b_H^L \equiv b_H^M$.

It is easy to show that the H-cost optimal path problem is equivalent to a standard optimal path problem in the lifted graph $\mathcal{G}_H$. To this end, define for $k \in \{H+1, \ldots, P\}$:

$$\tilde{g}_H(J_{k-1}, J_k) := \begin{cases} g_H(j_{k-H-1}, \ldots, j_k), & k < P, \\ g_H(j_{k-H-1}, \ldots, j_k) + H, & k = P. \end{cases}$$

Then $\sum_k \tilde{g}_H(J_{k-1}, J_k) = H + \sum_k g_H(j_{k-H-1}, \ldots, j_k) = \mathcal{J}_H(\pi)$, where $k \in \{H+1, \ldots, P\}$. The minimization of $\sum_k \tilde{g}_H(J_{k-1}, J_k)$ is a standard optimal path problem in the lifted graph $\mathcal{G}_H$.

C. Assignment of Transition Costs to Histories

We seek to assign transition costs on histories, i.e. values taken by the function g_H, such that some of the main aspects of the vehicle's motion capability (e.g. turn rate limits) influence the higher-level planning problem. Stated differently, the result of the higher-level task planner is a path $\pi^0 = \{j_0, \ldots, j_P\}$ in the graph $\mathcal{G}_0$, and we seek to assign transition costs to histories such that for any path in $\mathcal{G}_0$ with finite H-cost, there exists an admissible control $u \in \mathcal{U}_{[0,t_\text{f}]}$ and $t_\text{f} \in \mathbb{R}_+$ such that, for a given initial state $\xi_0 \in \mathcal{D}$,

$$\mathsf{x}(\xi(t; \xi_0, u)) \in \cup_{k=0}^{P} \mathsf{cell}(j_k), \qquad t \in [0, t_\text{f}]. \quad (2)$$

Equation (2) is a precise expression of the notion of "compatibility" between the higher- and lower levels in the proposed hierarchical motion-planning technique. Either of the following two approaches can be two adopted to assign transition costs to histories:

1) Assignments such that g_H and $\tilde{g}_h$ take values in the binary set $\{1, \chi\}$: the transition cost quantifies whether or not a history is feasible for traversal. Here $\chi \in \mathbb{R}_+$ is chosen sufficiently large, and a path with H-cost greater than χ is considered infeasible for traversal. We refer to such a transition cost function as *coarse*.
2) Assignments such that g_H and $\tilde{g}_H$ take values in $\mathbb{R}_+$: the transition cost provides a finer-grained merit of "goodness" of a history. We refer to such a transition cost function as *refined*. An example of such a cost function appears in [11], where a local trajectory optimization procedure finds specific vehicle trajectories that traverse the cells associated with a history. The cost of the history is assigned equal to the cost of this local trajectory.

An example of a coarse transition cost function is available in [24], which we present here for the sake of completion. Consider vertices $I, J \in V_H$ such that $(I, J) \in E_H$. We denote by $\mathcal{S}(I) \subset \mathcal{D}$ a set of states of the system Γ associated with $I \in V_H$, such that $\mathsf{x}(\mathcal{S}(I)) \subseteq \mathsf{cell}([I]_1) \cap \mathsf{cell}([I]_2)$. Thus, the position components of the states in $\mathcal{S}(I)$ lie on the boundary between the first and second cells associated with the ordered H-tuple I. Next let $\mathcal{Q}(J) \subset \mathcal{D}$ be a set of states such that $\mathsf{x}(\mathcal{Q}(J)) \subseteq \mathsf{cell}([J]_1) \cap \mathsf{cell}([J]_2)$ and for every state $\xi_0 \in \mathcal{Q}(J)$ there exists $t_\mathrm{f} \in \mathbb{R}_+$ and an admissible control input $u \in \mathcal{U}_{[0,t_\mathrm{f}]}$ such that $\mathsf{x}(\xi(t; \xi_0, u)) \in \bigcup_{k=1}^{H+1} \mathsf{cell}([J]_k)$, for all $t \in [0, t_\mathrm{f}]$. Informally, $\mathcal{Q}(J)$ is the set of all states whose position components lie on the boundary between the first and second cells associated with J, and such that the traversal of the geometric region defined by the cells associated with J is possible from any initial state within $\mathcal{Q}(J)$.

Next, let $\mathcal{R} : \mathcal{D} \to 2^{\mathcal{D}}$ be a reachability map associated with the sets $\mathcal{S}(I)$, defined by

$$\mathcal{R}(\xi_0) := \Bigg\{ \xi_t \in \mathcal{D} : \xi_t \in \bigcup_{t \in \mathbb{R}_+} \bigcup_{u \in \mathcal{U}_{[0,t]}} \bigcup_{\xi_0 \in \mathcal{S}(I)} \xi(t; \xi_0, u), \text{ and } \bigcup_{\tau \in [0,t]} \mathsf{x}(\xi(\tau; \xi_0, u)) \bigcap \overline{\mathsf{cell}([I]_2)} = \varnothing \Bigg\}, \tag{3}$$

where $\xi_0 \in \mathcal{S}(I)$ and $2^{\mathcal{D}}$ is the collection of all subsets of $\mathcal{D}$. Finally, for an element (I, J) of E_H, let $\hat{\mathcal{S}}(I, J) := \{\xi \in \mathcal{S}(I) : \mathcal{R}(\xi) \cap \mathcal{Q}(J) \neq \varnothing\}$. For a path $\pi = \{j_0, \ldots, j_P\}$ in $\mathcal{G}$, let $I_k := (j_{k-H-1}, \ldots, j_{k-1})$ for each $k \in \{H+1, \ldots, P\}$. Clearly, $(I_k, I_{k+1}) \in E_H$. The function $\tilde{g}_H$ and the set association $\mathcal{S}(\cdot)$ are iteratively defined as follows:

$$\mathcal{S}(I_{k+1}) := \bigcup_{\xi_0 \in \hat{\mathcal{S}}(I_k, I_{k+1})} \left(\mathcal{R}(\xi_0) \cap \mathcal{Q}(I_{k+1}) \right), \tag{4}$$

$$\tilde{g}_H(I_k, I_{k+1}) := \begin{cases} \chi, & \text{if } \mathcal{S}(I_{k+1}) = \varnothing, \\ 1, & \text{otherwise,} \end{cases} \tag{5}$$

with $\mathcal{S}(I_{H+1}) := \xi_{\mathrm{init}}$, where $\xi_{\mathrm{init}} \in \mathcal{D}$ is a prespecified initial state of the system Γ. Algorithms for computing the sets $\mathcal{S}(\cdot)$, $\mathcal{Q}(\cdot)$ and $\mathcal{R}(\cdot)$ for the Dubins car kinematic model (a model of a vehicle that moves forwards only at a constant speed with a fixed minimum turn radius) are provided in [25].

III. Incremental Path Repair

The proposed incremental planning algorithm is motivated by the following property of the transition cost function.

Proposition 1. *Let $\pi = \{j_0, \ldots, j_P\}$ be a path in the graph $\mathcal{G}$ such that $\mathcal{J}_H(\pi) < \chi$. Let $I_k^H \in V_H$ be vertices defined by $I_k^H := (j_{k-H}, \ldots, j_k)$, for $k \in \{H, \ldots, P\}$. Then for each $\ell \in \{H+2, \ldots, P\}$, either*

$$\tilde{g}_{H+1}(I_{\ell-1}^{H+1}, I_\ell^{H+1}) = \chi, \quad \text{or}$$
$$\tilde{g}_{H+1}(I_{\ell-1}^{H+1}, I_\ell^{H+1}) \leqslant \tilde{g}_H(I_{\ell-2}^H, I_{\ell-1}^H) + \tilde{g}_H(I_{\ell-1}^H, I_\ell^H).$$

If $\tilde{g}_H$, $H \geqslant 0$, are coarse transition cost functions, then the preceding inequality reduces to an equality:

$$\tilde{g}_{H+1}(I_{\ell-1}^{H+1}, I_\ell^{H+1}) = \tilde{g}_H(I_{\ell-2}^H, I_{\ell-1}^H) + \tilde{g}_H(I_{\ell-1}^H, I_\ell^H).$$

Proof. Omitted for lack of space. □

By definition, the sequence of vertices in V defined by the edge $(I_{\ell-1}^{H+1}, I_\ell^{H+1}) \in E_{H+1}$ is the same as that defined by the successive edges $(I_{\ell-2}^H, I_{\ell-1}^H), (I_{\ell-1}^H, I_\ell^H) \in E_H$. Also, for each $H \geqslant 0$ and $k \in \{H, \ldots, P\}$, the vertex I_ℓ^H belongs to the path $\pi^H = b_0^H(\pi)$. Proposition 1 states that either the edge $(I_{\ell-1}^{H+1}, I_\ell^{H+1})$ in graph $\mathcal{G}_{H+1}$ is infeasible for traversal, or the transition cost of this edge is no greater than the sum of the transition costs of the successive edges $(I_{\ell-2}^H, I_{\ell-1}^H)$ and $(I_{\ell-1}^H, I_\ell^H)$ in graph $\mathcal{G}_H$, for each $\ell \in \{H+2, \ldots, P\}$. Therefore, either $\mathcal{J}_{H+1}(\pi) \geqslant \chi$ or $\mathcal{J}_{H+1}(\pi) \leqslant \mathcal{J}_H(\pi)$.

Informally, this observation about $\mathcal{J}_{H+1}(\pi)$ means that a path π in $\mathcal{G}$, upon "further scrutiny" with higher values of the parameter H, will either be infeasible for traversal by trajectories of the vehicle dynamical system model Γ, or the cost of this traversal can potentially be reduced. Stated differently, Prop. 1 implies that is that it is desirable to use as large a value of the parameter H as possible, e.g. $H = \bar{H} = \mathrm{diam}(\mathcal{G})$ in the search for an optimal high-level plan. However, it is computationally impractical to do so, because the number of vertices in $\mathcal{G}_H$ increases exponentially with H.

To address this issue, we propose an incremental algorithm to search for a $\bar{H}$-cost optimal path in $\mathcal{G}$. The main idea is to seed the search for this path with a path that is optimal in the graph $\mathcal{G}_0$. The proposed algorithm uses a path repair procedure that replaces edges that are either infeasible or of high cost in lifted graphs. Starting with $H = 0$ (or any other small integer value of H that is convenient for computation of the initial path), the proposed algorithm incrementally progresses to higher values of H.

A precise description of the proposed incremental H-cost motion-planning algorithm is provided in Fig. 3. For simplicity of exposition, the algorithm in Fig. 3 considers coarse transition cost functions, and it can accommodate refined cost functions with no modifications in principle. In Section IV-B,

Incremental H-cost Motion-planning Algorithm

procedure MAIN

1: $H \leftarrow 0$, $n \leftarrow 0$
2: $\pi_0^0 \leftarrow \arg\min_{\pi \in \mathcal{L}_0^*(j_S, j_G)} \{\mathcal{J}_0(\pi)\}$
3: $H \leftarrow H + 1$
4: **while** $H \leq \bar{H}$ **do**
5: $P_n \leftarrow$ (number of vertices of π_n^0) -1
6: I_k^H, $k \in \{H, \ldots, P_n\}$ defined as in Prop. 1
7: **if** $\mathcal{J}_H(\pi_n^0) < \chi$ **then**
8: **for** $k = P_n - 1, P_n - 2, \ldots, 0$ **do**
9: **if** $\min_{\varsigma \in \mathcal{L}_H^*(I_k^H, I_{P_n}^H)} \{\mathcal{J}_H(\varsigma)\} < \sum_{\ell=k+1}^{P_n} \tilde{g}_H(I_{\ell-1}^H, I_\ell^H)$ **then**
10: $\pi_{n+1}^H \leftarrow$ path in $\mathcal{G}_H$ obtained by replacing with $\arg\min_{\varsigma \in \mathcal{L}_H^*(I_k^H, I_{P_n}^H)} \{\mathcal{J}_H(\varsigma)\}$ the edges between I_k^H and $I_{P_n}^H$ in path $b_0^H(\pi_n^0)$
11: $\pi_{n+1}^0 \leftarrow b_H^0(\pi_{n+1}^H)$
12: $n \leftarrow n + 1$
13: $H \leftarrow H + 1$
14: **else**
15: $k^* \leftarrow \arg\min\{k \in \{H+1, \ldots, P_n\} : \tilde{g}_H(I_{k-1}^H, I_k^H) = \chi\}$
16: $\{\varsigma^*, \ell^*\} \leftarrow$ PATH-REPAIR(k^*)
17: $\pi_{n+1}^H \leftarrow$ path in $\mathcal{G}_H$ obtained by replacing with ς^* the edges between $I_{k^*-1}^H$ and $I_{\ell^*}^H$ in path $b_0^H(\pi_n^0)$
18: $\pi_{n+1}^0 \leftarrow b_H^0(\pi_{n+1}^H)$
19: $n \leftarrow n + 1$

procedure PATH-REPAIR(k^*)

1: **for** $\ell = k^*, \ldots, P_n$ **do**
2: $\varsigma^* \leftarrow \arg\min\{\varsigma \in \mathcal{L}_H^*(I_{k^*-1}^H, I_\ell^H)\}$
3: **if** $\mathcal{J}_H(\varsigma^*) < \chi$ **then**
4: Return $(\varsigma^*, \ell^* = \ell)$

Fig. 3. Pseudocode description of the proposed incremental algorithm for solving the H-cost optimal path problem.

we discuss some aspects of practical implementations using refined transition costs. In Fig. 3, the algorithm is initialized in Lines 1–2 by searching for a "seed" path in graph $\mathcal{G}_0$ with prespecified initial and goal vertices $j_S, j_G \in V$. The result of the algorithm at the n^{th} iteration, where $n \in \mathbb{Z}_+$, is a path π_n^0 in $\mathcal{G}$ from the initial vertex j_S to the goal vertex j_G. Lines 4–19 describe the iterative process of the algorithm as it progresses through increasing values of H. Whenever H is incremented and an infeasible edge in detected in $\mathcal{G}_H$, the algorithm replaces this infeasible edge with a subpath with finite H-cost, as described in Lines 15–19. The path repair procedure in Line 16 computes this subpath from the vertex immediately before the infeasible edge to any of the vertices between the infeasible edge and the goal. In Lines 8–11, the algorithm attempts to find paths of lower cost between intermediate vertices of the currently known path and the goal vertex. This step is necessary because the "patching" of subpaths in Line 17 can lead to suboptimal paths. However, in practical implementations, for the sake of speed Lines 9-11 need be executed only for high values of H, or they can be executed after completing the `while` loop of Line 4.

The optimization involved in Line 2 of the PATH-REPAIR procedure can be performed by an algorithm similar to the H-cost optimal path-planning algorithm described in [11]. Because the number of vertices in $\mathcal{G}_H$ separating $I_{k^*-1}^H \in V_H$ and $I_\ell^H \in V_H$ is small, the search for a subpath between these vertices is fast.

The pseudocode provided in Fig. 3 omits for the sake of simplicity some obvious exceptions. For example, if no path with H-cost less than χ exists in the graph $\mathcal{G}$, then the procedure PATH-REPAIR will be unable to find a subpath ς^*, and the algorithm resolves this exception by reporting failure.

The result of this algorithm is the path returned at either Line 11 or Line 18 of the final iteration of the algorithm. Whereas the algorithm terminates after a finite number of iterations (as we show next), its execution can be forcibly halted after Line 19 at any an intermediate iteration n before the algorithm's natural termination. Crucially, despite such a forcible termination, the algorithm returns a path π_n^0 in graph $\mathcal{G}$ which is feasible for traversal (i.e. has H-cost less than χ), assuming such a path exists. This property of the algorithm of having ready a feasible path at the end of any iteration is highly desirable in real-time application, especially in light of the fact that the algorithm *eventually* finds a $\bar{H}$-cost optimal path.

The following result further highlights the merits of the proposed algorithm: not only does the algorithm eventually converge to an $\bar{H}$-cost optimal path, but the H-costs of paths found in intermediate iterations are nonincreasing. Whenever H is incremented (Line 13), there is a possibility that the transition costs of some edges in $\mathcal{G}_{H+1}$ become higher than χ, and therefore the trend nonincreasing costs can be interrupted.

Proposition 2. *For a fixed value of the parameter H, whenever Lines 8–12 of Fig. 3 are executed, $\mathcal{J}_H(\pi_{n+1}^0) \leqslant \mathcal{J}_H(\pi_n^0)$, for each $n \in \mathbb{Z}_+$.*

Proof. Immediately obvious by construction of π_{n+1}^0 in Lines 10–11 of Fig. 3. □

Next, we state the main result of this paper.

Proposition 3. *The proposed algorithm as described in Fig. 3 terminates after a finite number of iterations. Upon termination, the following statements hold true:*

1) *If, for some $\tilde{H} \leqslant \bar{H}$, there exists no path in graph $\mathcal{G}$ with $\tilde{H}$-cost less than χ, then the algorithm reports failure.*
2) *If there exists at least one path in graph $\mathcal{G}$ with $\bar{H}$-cost less than χ, then the algorithm returns a path with minimum $\bar{H}$-cost.*

Proof. Every iterative loop in the proposed algorithm in Fig. 3 has a finite termination condition. Therefore the overall algorithm must terminate in a finite number of iterations.

Case 1) Suppose for some $\tilde{H} \leqslant \bar{H}$, there exists no path in graph $\mathcal{G}$ with H-cost less than χ and assume (without loss of generality) that $\tilde{H}$ is the smallest such value of the

parameter H. Suppose that the algorithm increments the value of the parameter H from $\tilde{H}-1$ to $\tilde{H}$ at the n^{th} iteration. Then $\mathcal{J}_{\tilde{H}}(\pi_n^0) \geqslant \chi$, and in the next iteration the algorithm executes Lines 7 and 15. At Line 15, the PATH-REPAIR must report failure because a subpath satisfying the condition in Line 3 of the PATH-REPAIR procedure does not exist. Then the overall algorithm also reports failure.

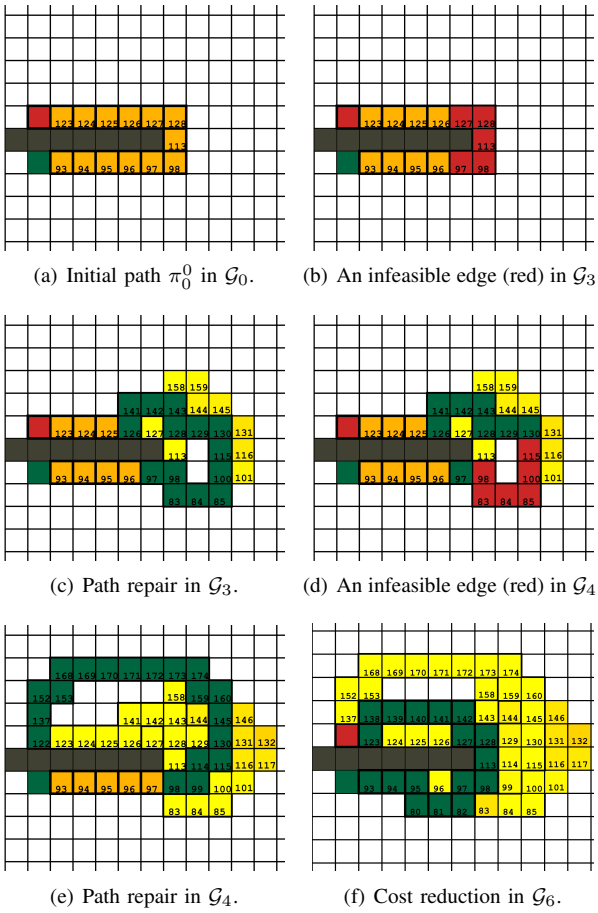
Case 2) Suppose there exists at least one path in graph $\mathcal{G}$ with $\bar{H}$-cost less than χ. Then there exists a path in graph $\mathcal{G}$ with minimum $\bar{H}$-cost, because the number of such paths is finite, in turn because the numbers of vertices and edges in $\mathcal{G}$ are finite. According to the termination condition in Line 4, the overall algorithm terminates when $H = \bar{H}$. For any particular value of H, the final iteration of the `for` loop in Line 8 results in an H-cost optimal path. In particular, it results in the $\bar{H}$-cost optimal path. □

Remark: It is important to note that the proposed algorithm does not reduce the complexity of the H-cost optimal path problem, as is evident from the proof of Prop. 3. However, the proposed algorithm ensures that the search for the $\bar{H}$-cost optimal path does not proceed in a "all or nothing" manner. Specifically, it makes available a feasible solution during intermediate iterations. Also, in practice, the `for` loop in Line 8 will terminate much earlier than the stated termination condition of $k = 0$. As evident from the preceding proof, the execution of Lines 8–12 can be executed only for $H = \bar{H}$, without affecting the truth of Prop. 3.

IV. ILLUSTRATIVE EXAMPLES AND DISCUSSION

Figure 4 illustrates the execution of the proposed algorithm. Here, the prespecified initial and goal vertices are indicated, respectively, by the green- and red-colored vertices in Fig. 4(a). The path between the initial and final vertices highlighted in Fig. 4(a) is a 0-cost optimal path, i.e., it is found by solving the standard optimal path problem on $\mathcal{G}_0$, where edge transition costs are defined as the Euclidean distances between cells. The algorithm progresses to higher values of H, and when $H = 3$, the 3-cost of the path is found (Line 7 of the algorithm in Fig. 3) to be greater than or equal to χ. In particular, the transition cost of a single edge, illustrated in red in Fig. 4(b) is found to be equal to χ. In this example, H-costs are defined by Eqns. (4) and (5) for a Dubins car kinematic model with unit minimum turn radius. The edge with transition cost χ is replaced by the PATH-REPAIR procedure. The subpath ς^* computed by the PATH-REPAIR procedure is indicated in green color in Fig. 4(c). Next, the 4-cost of the new path is found to be greater than or equal to χ (Fig. 4(d)) and it is again replaced by a subpath by the PATH-REPAIR procedure (Fig. 4(e)). The repaired path is found to have 6-cost less than χ, and at $H = 6$, the cost reduction steps in Lines 8-11 of the algorithm in Fig. 3 are executed to arrive at a 6-cost *optimal* path as shown in Fig. 4(f). The yellow-colored cells in Fig. 4 indicate the vertices explored by the PATH-REPAIR procedure during its computations to find a subpath (Line 16 in Fig. 3).

Figure 6 illustrates a more striking example of application the proposed algorithm. Here, a 4-cost feasible path is found quickly, and in further iterations, the algorithm finds paths of lower 4-cost. A feasible path is provided during intermediate iterations, and an optimal is eventually found.



(a) Initial path π_0^0 in $\mathcal{G}_0$. (b) An infeasible edge (red) in $\mathcal{G}_3$.

(c) Path repair in $\mathcal{G}_3$. (d) An infeasible edge (red) in $\mathcal{G}_4$.

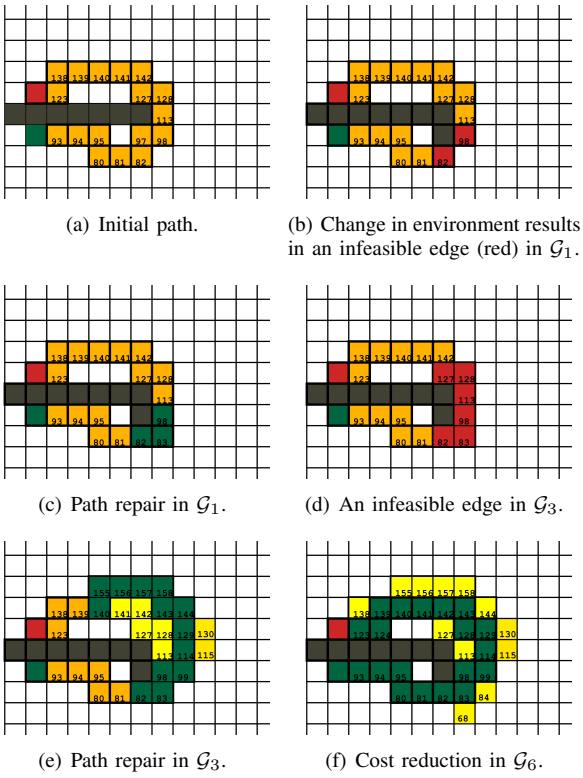
(e) Path repair in $\mathcal{G}_4$. (f) Cost reduction in $\mathcal{G}_6$.

Fig. 4. Illustrative example of solution of the H-cost optimal path problem using incremental path repair.

A. Application to Replanning with Environmental Changes

In the geometric path-planning literature, anytime incremental algorithms such as LPA* [26] have been discussed for fast replanning of shortest (by Euclidean distance) paths when small changes in the environment are detected. These changes can result due to, say, moving obstacles or improved perception of the environment. The proposed incremental algorithm serves the purpose of replanning H-cost optimal paths in response to such changes in the environment. Specifically, when coarse transition costs are considered, the current H-cost feasible or optimal path can become infeasible (i.e., its cost can increase beyond χ) if changes in the environment cause a cell in the current path to be blocked by an obstacle. The proposed algorithm can detect such a change in Line 7 of Fig. 3, and the rest of the repair procedure is executed as before.

Figure 5 illustrates an example of this replanning application of the proposed algorithm. The initial and goal vertices are indicated in Fig. 5(a) by green- and red-colored cells. The black-colored cells indicate obstacles. As illustrated in



(a) Initial path. (b) Change in environment results in an infeasible edge (red) in $\mathcal{G}_1$. (c) Path repair in $\mathcal{G}_1$. (d) An infeasible edge in $\mathcal{G}_3$. (e) Path repair in $\mathcal{G}_3$. (f) Cost reduction in $\mathcal{G}_6$.

Fig. 5. Illustrative example of replanning in response to changes in the environment.

Fig. 5(b), a change in the location of obstacles makes infeasible for traversal the initial path π_0^0 illustrated in Fig. 5(a). In Fig. 5(a), the algorithm has already progressed to $H = 1$, and due to the change in the environment, the 1-cost of the initial path is found to be higher than χ. The PATH-REPAIR procedure finds a subpath, indicated in green in Fig. 5(c). In a later iteration, the 3-cost of the new path is found higher than χ; in particular one edge (indicated in red in Fig. 5(d)) is found to have cost χ. The PATH-REPAIR procedure finds a subpath in $\mathcal{G}_3$ to replace this edge, as indicated by the green-colored subpath in Fig. 5(e). The repaired path is found to have 6-cost less than χ and at $H = 6$, the cost reduction steps in Lines 8-11 of Fig. 3 are executed to arrive at a 6-cost optimal path as shown in Fig. 5(f).

Notice that in each of the two preceding examples, a 6-cost[1] *feasible* path is available in iterations *before* the optimal path is found. In real-time applications, this property can be used to enforce hard bounds on the computation time: the proposed algorithm will report a *feasible* path in the available computation time. By Prop. 3, if the algorithm is allowed sufficient computation time, it will converge to an optimal path, and by Prop. 2, the resultant H-costs are nonincreasing in all intermediate iterations except, possibly, in iterations where H is incremented.

[1]In these particular examples, 6-cost feasible paths were found to be feasible for all higher values of H.

B. Further Discussion of the Proposed Algorithm

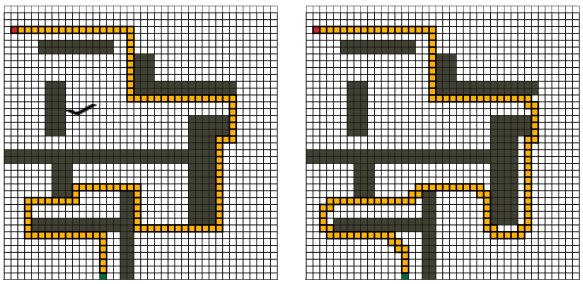
As previously stated, refined transition costs can be assigned to H-histories, for example by solving a local trajectory optimization problem. The proposed algorithm as described in Fig. 3 is applicable, in principle, when refined transition cost functions are used. In practical implementations, however, the cost reduction steps in Lines 8–12 will slow down the overall algorithm because a large number of alternatives for cost reduction may be available, especially when the original graph $\mathcal{G}$ has a large number of vertices. To alleviate this problem, two modifications can be made to the algorithm in Fig. 3 without changing its theoretical properties. First, Lines 8–12 can be programmed to execute only for certain high values of H larger than a user-specified threshold. Second, Lines 8–12 can be programmed to execute only if the cost of edges to be replaced is higher than a certain user-specified threshold $\varepsilon \geqslant 0$. Specifically, Line 9 in Fig. 3 can be replaced by the following:

if $\min_{\varsigma \in \mathcal{L}_H^*(I_k^H, I_{P_n}^H)} \{\mathcal{J}_H(\varsigma)\} < \sum_{\ell=k+1}^{P_n} \tilde{g}_H(I_{\ell-1}^H, I_\ell^H)$ and $\tilde{g}_H(I_k, I_{k+1}) \geqslant \varepsilon$ **then**

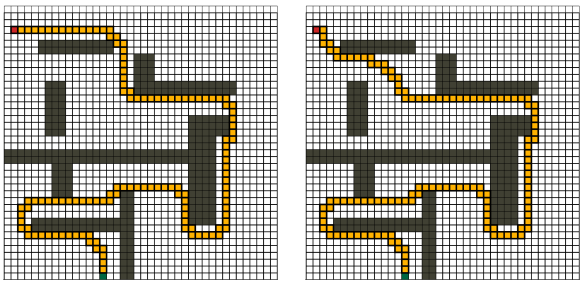
The proposed algorithm retains all of the merits of the H-cost motion-planning technique discussed in [11]. Therein, thorough comparisons of the H-cost technique with other motion-planning techniques including randomized sampling-based techniques are available. Furthermore, in [27], the H-cost motion-planning technique was implemented with multiresolution cell decompositions. Because the proposed algorithm does not affect the type of cell decompositions used to construct the graph $\mathcal{G}$, multiresolution implementations of the proposed algorithm can be easily developed in a manner similar to that discussed in [27].

V. CONCLUSIONS AND FUTURE WORK

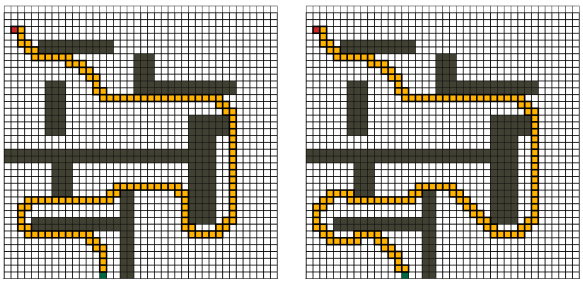
In this paper, we discussed an incremental algorithm for hierarchical motion-planning based on the previously developed H-cost motion-planning technique. The H-cost technique has been shown to be useful in the tight integration of the solutions of a high-level discrete task-planning problem with a low-level continuous trajectory generation problem. Whereas it is beneficial to use high values of the parameter H, the complexity of the H-cost optimal path problem increases exponentially with H. The proposed algorithm alleviates this high computational cost by making available a *feasible* solution at intermediate iterations We proved that the proposed algorithm is guaranteed to converge to an optimal solution given enough computation time (i.e. after a sufficiently large number of iterations). Furthermore, the cost of the solutions available in intermediate iterations of the proposed algorithm is always nonincreasing except, possibly, at a finite number of special iterations (i.e. when H is incremented). We illustrated the proposed algorithm using numerical simulation examples. Future work includes the extension of the proposed algorithm to the solution of more general task-planning problems, e.g. high-level specifications described using logic formulae.



(a) Initial path. (b) $H = 2$-cost feasible path.



(c) $H = 4$-cost feasible path. (d) Reduced 4-cost feasible path.



(e) Reduced 4-cost feasible path. (f) 4-cost optimal path.

Fig. 6. Illustrative example: during intermediate iterations, feasible solutions are available, whereas the solution cost is reduced.

Acknowledgments: The research reported in this paper was supported in part by startup funds provided by WPI to the second author, and in part by US Air Force Small Business Innovation Research (SBIR) contract #FA8501-14-P-0034, WPI sub-contract #AFS14-1186 from Aurora Flight Sciences Corp., Cambridge MA. We thank Jeffrey T. Chambers, Ph.D., Program Manager at Aurora.

REFERENCES

[1] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion," *IEEE Robotics and Automation Magazine*, pp. 61 – 70, March 2007.

[2] Z. Shiller and Y.-R. Gwo, "Dynamic motion planning of autonomous vehicles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 241–249, 1991.

[3] D. Zhu and J.-C. Latombe, "New heuristic algorithms for efficient hierarchical path planning," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 9–20, 1991.

[4] S. R. Cunha, A. C. de Matos, and F. L. Pereira, "An automatic path planning system for autonomous robotic vehicles," in *Proceedings of the IECON '93 International Conference on Industrial Electronics, Control, and Instrumentation*, (Maui, HI, USA), pp. 1442–1447, November 1993.

[5] J.-P. Laumond, M. Taix, P. Jacobs, and R. M. Murray, "A motion planner for nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 577–593, 1994.

[6] M. Cherif, "Kinodynamic motion planning for all-terrain wheeled vehicles," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, (Detroit, MI.), pp. 317 – 322, May 1999.

[7] D. Coombs, K. Murphy, A. Lacaze, and S. Legowik, "Driving autonomously offroad up to 35 km/h," in *Proceedings of the 2000 International Vehicles Conference*, 2000.

[8] A. Rosiglioni and M. Simina, "Kinodynamic motion planning," in *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, vol. 3, pp. 2243–2248, 2003.

[9] B. Mettler and E. Bachelder, "Combining on- and offline optimization techniques for efficient autonomous vehicle's trajectory planning," in *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference 2005*, vol. 1 of *499–511*, 2005.

[10] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, pp. 224–233, Mar–Apr 1985.

[11] R. V. Cowlagi and P. Tsiotras, "Hierarchical motion planning with dynamical feasibility guarantees for mobile robotic vehicles," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 379 – 395, 2012.

[12] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, pp. 971–984, July 2000.

[13] P. Tabuada, "Controller synthesis for bisimulation equivalence," *Systems and Control Lett.*, vol. 57, pp. 443–452, June 2008.

[14] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, pp. 864 – 874, October 2005.

[15] G. E. Fainekos, A. Girard, and G. J. Pappas, "Hierarchical synthesis of hybrid controllers from temproal logic specifications," in *Hybrid Systems: Computation and Control* (A. Bemporad, A. Bicchi, and G. Buttazzo, eds.), LNCS 4416, pp. 203 – 216, 2007.

[16] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, pp. 287–297, February 2008.

[17] X.-C. Ding, M. Lazar, and B. C, "Formal abstraction of linear systems via polyhedral Lyapunov functions," in *Proceedings of the 4th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS)*, (Eindhoven, The Netherlands), Jun 6–8 2012.

[18] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 396–409, 2012.

[19] X. C. Ding, M. Lazar, and C. Belta, "Receding horizon temporal logic control for finite deterministic systems," in *Proceedings of the 2012 American Control Conference*, (Montréal, Canada), pp. 715–720, Jun 27–29 2012.

[20] M. Svorenova, I. Cerna, and C. Belta, "Optimal receding horizon control for finite deterministic systems with temporal logic constraints," in *Proceedings of the 2013 American Control Conference*, (Washington, DC, USA), pp. 4399 – 4404, Jun 17–19 2013.

[21] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 95, pp. 1652–1659, 1995.

[22] T. Wongpiromsarn, V. G. Rao, and R. D. D'Andrea, "Two approaches to dynamic refinement in hierarchical motion planning," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, (San Francisco, CA, USA), Aug 15–18 2005.

[23] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. Athena Scientific, Belmont, MA, 2000.

[24] R. V. Cowlagi and D. N. Kordonowy, "Geometric abstractions of vehicle dynamical models for intelligent autonomous motion," in *Proceedings of the 2014 American Control Conference*, (Portland, OR.), pp. 4840–4845, Jun 4 – 6 2014.

[25] R. V. Cowlagi and P. Tsiotras, "Curvature-bounded traversability analysis for motion planning of mobile robots," *IEEE Transactions on Robotics*, 2014. in press.

[26] S. Koenig, M. Likhachev, Y. Liu, and D. Furcy, "Incremental heuristic search in AI," *Artificial Intelligence Magazine*, vol. 25, pp. 99–112, 2004.

[27] R. V. Cowlagi and P. Tsiotras, "Multi-resolution motion planning for autonomous agents via wavelet-based cell decompositions," *IEEE Transactions on Systems, Man and Cybernetics: Part B - Cybernetics*, vol. 42, no. 5, pp. 1455–1469, 2012.