T*ε—Bounded-Suboptimal Efficient Motion Planning for Minimum-Time Planar Curvature-Constrained Systems

Doron Pinsky* Petr Váňa† Jan Faigl† Oren Salzman‡

***Abstract*—We consider the problem of finding collision-free paths for curvature-constrained systems in the presence of obstacles while minimizing execution time. Specifically, we focus on the setting where a planar system can travel at some range of speeds with unbounded acceleration. This setting can model many systems, such as fixed-wing drones. Unfortunately, planning for such systems might require evaluating many (local) time-optimal transitions connecting two close-by configurations, which is computationally expensive. Existing methods either pre-compute all such transitions in a preprocessing stage or use heuristics to speed up the search, thus foregoing any guarantees on solution quality. Our key insight is that computing all the time-optimal transitions is both (i) computationally expensive and (ii) unnecessary for many problem instances. We show that by finding bounded-suboptimal solutions (solutions whose cost is bounded by $1+\varepsilon$ times the cost of the optimal solution for any user-provided ε) and not time-optimal solutions, one can dramatically reduce the number of time-optimal transitions used. We demonstrate using empirical evaluation that our planning framework can reduce the runtime by several orders of magnitude compared to the state-of-the-art while still providing guarantees on the quality of the solution.**

***Index Terms*—Motion and Path Planning, Nonholonomic Motion Planning, Constrained Motion Planning.**

I. Introduction

In this work, we study the problem of finding minimal-time collision-free paths for curvature-constrained systems with variable speed. Curvature constraints are prevalent in a variety of systems (see, e.g., [1], [2], [3], [4]). Unfortunately, determining whether a collision-free curvature-constrained path exists is NP-hard even for a planar system [5]. As a result, this continuous problem can be discretized into a graph data structure using sampling-based [6] or search-based approaches [7], which is then queried to obtain a discrete path representing a curvature-constrained solution in the continuous space. The graph's vertices correspond to robot configurations (i.e., d-dimensional points that uniquely describe the robot's position and orientation) and edges correspond to local motions taken by the robot.

Manuscript received: September 9, 2021; Revised December 18, 2021; Accepted January 17, 2022. This paper was recommended for publication by Editor Hanna Kurniawati upon evaluation of the Associate Editor and Reviewers' comments.

*Pinsky, D. is with the Technion Autonomous Systems Program (TASP), Technion, Haifa, Israel. E-mail: doron.pinsky@campus.technion.ac.il

‡Salzman, O. is with the Department of Computer Science, Technion, Haifa, Israel. E-mail:osalzman@cs.technion.ac.il

†Váňa, P. and Faigl, J. are with the Faculty of Electrical Engineering, Czech Technical University, 166 27 Prague, Czech Republic. E-mails: {vanapet1,faiglj}@fel.cvut.cz

This research was supported by grant No. 3-16079 from the Ministry of Science & Technology, Israel and by the Ministry of Education Youth and Sports (MEYS) of the Czech Republic under project No. LTAIZ19013.

Digital Object Identifier (DOI): see top of this page.

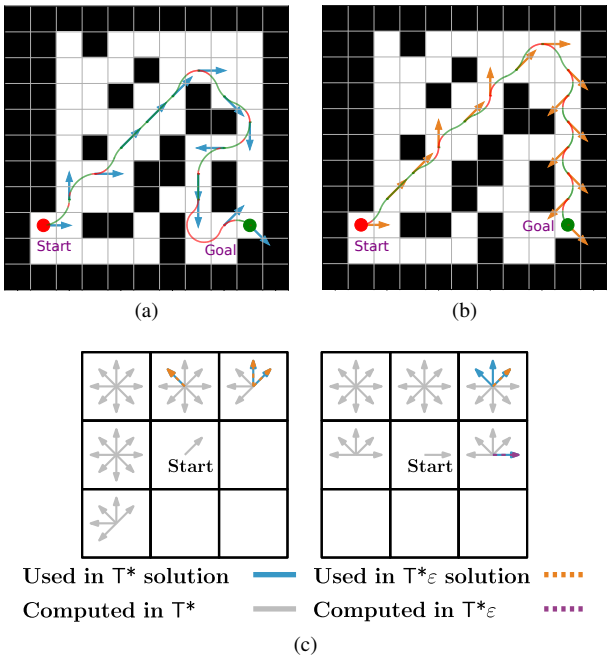


Fig. 1. (a), (b) The time-optimal and bounded-suboptimal path obtained by T* [8] and by T*ε using an approximation factor of $\varepsilon = 2$, respectively. The red and green segments correspond to settings where the system travels at the minimum and maximum speed, respectively. The colored arrows show the orientation of each configuration in the solution. T*ε finds a solution whose cost is 7 % larger than the cost of the optimal solution found by T*, but the runtime of T*ε is faster by a factor of roughly 13×. (c) Given a start orientation (diagonal on the left or horizontal on the right) in an eight-connected grid, there are 68 unique transitions in total (after taking into account symmetry and rotation) to the adjacent grid cells. The speedup of T*ε is obtained by computing only a small subset of the time-optimal transitions that are all computed by T*. Here, T*ε computes only five transitions, four of which are used in the found path. Figure best viewed in color.

Interestingly, the computational bottleneck in these search algorithms is a frequent computation of local time-optimal transitions between neighboring vertices, obtained using numerical optimization. Our key insight, depicted in Fig. 1, is that computing all the time-optimal transitions is both (i) computationally expensive and (ii) unnecessary for many problem instances. We introduce a novel algorithmic framework, which we call T*ε, that allows finding bounded-suboptimal solutions while reducing planning times by orders of magnitude compared to the state-of-the-art.

Our framework consists of three algorithmic components. The first component assumes that we have an efficient-to-compute *lower bound* on the cost of the optimal transition between two configurations. In Sec. VI, we demonstrate two such bounds—when the environment does not and does contain dynamics such as wind currents. The second algorithmic component is the A*ε-based search algorithm [9] that uses

these lower bounds on optimal transitions, together with a user-provided approximation factor ε, to choose which edges to consider. Roughly speaking, the search attempts to use only transitions for which the true (computationally expensive) cost was computed while guaranteeing the bound on the quality of the solution obtained. Finally, the third component is a heuristic approach to pre-compute a small set of transitions that are likely to be used in an optimal path.

Using efficient-to-compute lower bounds on the cost of local transitions is a common technique to speed up motion-planning algorithms (see, e.g., [10], [11], [12], [13], [14]). These are used to minimize the computation time taken to compute the cost of an edge or transition, also known as an *edge evaluation*. Typically, edge evaluation corresponds to computing if a robot intersects an obstacle while performing some local motion (also known as collision detection [15]). Thus, the computationally expensive operation of edge evaluation is applied to each edge individually. Indeed, it can be shown that under some mild assumptions, these approaches allow minimizing the number of edges evaluated [16].

In our case, we plan in a discretized space; thus, the setting is somewhat different. There is a fixed set of transitions that can be taken from *any* given configuration. Since these transitions are computationally expensive to compute, it is natural to try and *re-use* transitions that have already been computed. The challenge is how this should be done while ensuring that the cost of the path found is within a given multiplicative bound of the cost of an optimal path.

As we demonstrate in simulation (Sec. VI), our planning framework allows to compute only a small fraction of transitions, which, in turn, allows us to reduce the runtime by several orders of magnitude compared to the state-of-the-art, especially in dynamic conditions.

II. Related Work

We now continue to review related work on planning for minimum-time planar curvature-constrained systems. When the system is constrained to travel at a single speed, an optimal path connecting two configurations (i.e., two planar locations, each associated with its angular heading) can be computed analytically [17]. In this setting, an optimal path, also known as *Dubins path*, is one of six types: RSR, RSL, LSR, LSL, RLR, and LRL, where R and L refer to right and left turns, respectively, and S refers to going straight. As the system travels at a single speed, the shortest and the time-optimal paths are identical.

The kinematic model considered in this work is the setting where the system can travel at some range of speeds but with unbounded acceleration (i.e., transitioning between low and high-speed can be done instantaneously). It is not hard to see that, in such a setting, the shortest path is attained by computing the optimal Dubins path, assuming that the system travels at the minimum speed (as the system can travel using the smallest turning radius possible and can thus better maneuver). Computing the time-optimal path requires alternating between low-speed (to allow for tighter turns) and high-speed motions (to allow for faster progress).

Wolek *et al.* [18] showed that it is sufficient to consider only the two extreme speeds and identified a sufficient set of 34 candidate paths (in contrast to the six-candidate paths when the system travels at a single speed). Each candidate path contains circular arcs and straight-line segments on which the vehicle can travel at either of the two extreme speeds. However, in contrast to Dubins paths, which can be computed analytically, computing these time-optimal paths for a variable-speed system requires a numerical optimization that is (i) much more computationally expensive and (ii) may return locally optimal solutions (and not globally optimal ones). Kučerová *et al.* [19] showed an efficient heuristic approach to find high-quality paths when considering time as the cost function by using multiple turning radii. However, there is no guarantee regarding the quality of these paths.

For each model mentioned above, motion-planning algorithms that account for environmental obstacles were introduced. These include both sampling-based approaches such as the work by Wilson *et al.* [20], search-based methods such as the work by Song *et al.* [8], and hybrid methods that borrow ideas from both motion-planning disciplines [21]. Of specific interest to the presented work is T* [8], a time-optimal risk-aware motion-planning algorithm that obtains a time-optimal solution but requires a time-consuming preprocessing phase. As our work builds upon the algorithmic foundation of T*, we describe it in Sec. IV. Following T*, Wilson *et al.* [22] developed a fast motion-planning algorithm that uses Dubins paths of various speeds to reduce the computational effort and runtime of T*. For the settings evaluated, the costs of solutions obtained by this algorithm are near-optimal, but there are no guarantees regarding the quality of solutions.

The aforementioned kinematic model can be extended to account for external dynamic changes such as wind or ocean currents. For such systems, a minimum-time path can be computed similarly to the setting where no wind exists [23]. However, only two candidate path types have analytic solutions (RSR and LSL). Mittal *et al.* [24] used those two types to compute a path between two vehicle poses under ocean currents. These kinematic models were used not only in the context of single-goal motion planning problems but also in the settings where the objective is to reach multiple goals [25] and further extended to account for some notion of rewards [26].

III. Problem Statement

Our problem formulation follows Song *et al.* [8]. Specifically, we assume a variable-speed curvature-constrained planar robotic system with unbounded acceleration. The system's dynamics can be described by

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{pmatrix} = \begin{pmatrix} v(t)\cos\theta(t) \\ v(t)\sin\theta(t) \\ u(t) \end{pmatrix}. \tag{1}$$

Here, $(x, y, \theta) \in \mathrm{SE}(2)$ is the robot's placement and orientation, v is the robot's speed (which is considered as a control input) and u is the second control input dictating the

system's turning rate via maximal lateral acceleration K that is determined for the specific system[1] as

$$|u| \leq \frac{K}{v}. \tag{2}$$

The speed is limited by some minimum and maximum values denoted by $v_{\min}$ and $v_{\max}$, respectively. W.l.o.g., we assume that $v_{\max} = 1$. Thus, the system is constrained to travel between two extreme radii:

$$\rho_{\max} = \frac{v_{\max}^2}{K}, \quad \text{and} \quad \rho_{\min} = \frac{v_{\min}^2}{K}. \tag{3}$$

We assume that the continuous workspace is discretized into a grid according to a predefined resolution. Each cell can be categorized as *free* or *forbidden* corresponding to locations that the system can and cannot occupy, respectively. In addition, we assume that every robot motion starts and ends at the center of a cell. Specifically, at each step, the robot transitions to one of its eight neighbors and with one of eight possible orientations.

A *path* γ is a sequence of configurations where the transition between them obeys the system's dynamics and constraints (Eq. 1). It is said to be *collision free* if it only occupies free cells. The cost $c(\gamma)$ of a path γ is the step-wise cost of the transitions between any consecutive configurations in γ. It can be computed using the system velocity along γ. Namely,

$$c(\gamma) = \int_{\gamma} \frac{1}{v(\tau)} d\tau, \tag{4}$$

where $v(\tau)$ refers to the speed along the path segment $d\tau$.

Given start and goal configurations $s_{\text{start}}, s_{\text{goal}} \in \text{SE}(2)$, a collision-free path γ is said to be *optimal* if it (i) starts at s_{start} and ends at s_{goal}; and (ii) there is no path γ' connecting s_{start} and s_{goal} such that $c(\gamma') < c(\gamma)$. Similarly, it is said to be *bounded-suboptimal* for some approximation factor $\varepsilon \geq 0$ if it (i) starts at s_{start} and ends at s_{goal}; and (ii) for any path γ' connecting s_{start} and s_{goal}, $c(\gamma) \leq (1+\varepsilon) \cdot c(\gamma')$.

While previous works (see, e.g., [8]) were concerned with finding optimal paths, in this work, we are interested in finding bounded-suboptimal paths for a user-provided approximation factor ε. As we will see, the extra flexibility obtained by finding bounded-suboptimal paths allows us to reduce running times by orders of magnitude with little compromise on the path quality in practice.

IV. Algorithmic Background

As both A*ε [9] and T* [8] serve as the algorithmic foundation of our work, we provide a brief description of these two algorithms.

A. A*ε (A-star epsilon)

A*ε (also known as FOCAL search) [9] is a bounded-suboptimal search algorithm based on the celebrated A* algorithm [27]. Like A*, it searches a graph by continuously expanding nodes, starting from the start node until the goal node is reached. To ensure optimality, A* orders nodes in a priority list called OPEN. Nodes in the OPEN list are ordered according to their f-value that is the sum of the cost to reach the node from the source (also known as the cost-to-come or g-value and denoted by $g(n)$ for a node n) added to a conservative estimate of the cost to reach the goal (also known as the cost-to-go or h-value and denoted by $h(n)$ for a node n). Namely for a node n, its f-value is

$$f(n) = g(n) + h(n). \tag{5}$$

Unlike A*, A*ε also uses a so-called FOCAL list containing all nodes from the OPEN list whose f-values are no larger than $1+\varepsilon$ times the smallest f-value in the OPEN list. It can be shown that expanding any sequence of nodes from the FOCAL list ensures that the cost of the final solution will indeed be bounded by a factor of $1+\varepsilon$ when compared to the cost of the optimal solution.

B. T* (T-star)

T* [8] builds on the approach by Wolek *et al.* [18] that allows finding an optimal[2] solution for the time-optimal transition between any two configurations in an obstacle-free space. These transitions are used to compute all possible motions for an agent in a discretized eight-connected grid. In this space, there is a finite set of possible transition types. For example, a transition can be moving from configuration $(0,0,0)$ to configuration $(1,1,\pi/2)$, which corresponds to moving diagonally in the NE direction while changing the system's heading.

To this end, there are a total of $8 \times 8 \times 8 = 512$ possible transitions (eight possible start and target orientations and eight neighboring cells). However, after accounting for symmetry and rotation, there are only 68 unique transitions.

After pre-computing all possible transitions, an A*-like search is used to find a minimal-cost path. In their original work, Song *et al.* [8] consider both minimal-time paths and a cost-function that balances the time and the risk of colliding with an obstacle. However, we consider only minimum-time paths in this work and refer to T* as the search algorithm that optimizes this criterion.

It is important to note that in many settings it is not possible to pre-compute all transitions in an offline phase. This is because optimization-related parameters might be available only when a query is provided. For example, in the presence of wind conditions, which affect the system's dynamics and hence the transitions' computation, wind parameters are only available to the planner when a query is provided. For further details, see Sec. VI.

Interestingly, after extensive empirical evaluation, we noticed that: (i) pre-computing all possible transitions last two orders of magnitude more than the A*-like search; (ii) only a small fraction of all possible time-optimal transitions are part of an optimal path found by T*; and (iii) Dubins path computed using the minimal speed $v_{\min}$ (whose cost is a lower bound

[1]For example, $K = g \tan \varphi_{\max}$ for fixed-wing vehicles, where g is the gravitational acceleration and $\varphi_{\max}$ the maximum allowed bang angle.

[2]To be precise, the solutions computed are only an approximation of the time-optimal solutions since the computation is based on numerical optimizations, which may not converge to a global optimum. Thus, the optimality of T* and the bounded-suboptimality of T*ε is only with respect to (w.r.t.) the quality of the local-optimization.

Algorithm 1: T*ε

Input: map, s_{start}, s_{goal}, approximation factor ε
Output: bounded-suboptimal path connecting s_{start} to s_{goal}

S1. compute $c(\cdot)$ for transitions in shortest path
1 $\gamma_{\text{SP}}^{v_{\min}} \leftarrow$ minimal-length collision-free Dubins path on map connecting s_{start} to s_{goal} using $v_{\min}$
2 **for each** transition $T \in \gamma_{\text{SP}}^{v_{\min}}$ **do**
3 compute $c(T)$, cache result and set $\kappa(T) = 0$

S2. A*ε-like search
4 run A*ε from n_{start} (node with state s_{start}) with OPEN and FOCAL ordered according to Eq. 5 and 7, respectively
5 **when** expanding a node n **do**
6 **if** incoming transition $n.T$ was not computed ($\kappa(n.T) = 1$) **then**
7 compute $c(n.T)$, cache result, set $\kappa(n.T) = 0$
8 **continue**
9 **if** configuration associated with the node n is s_{goal} **then**
10 **return** path associated with n
11 **for each** succesor node n' of n in map **do**
12 use Eq. 6 to evaluate the g-value of n'

on the length of the time-optimal path) is often in the same homotopy class as the optimal path and shares a significant portion of its transitions. These observations are key to our algorithmic framework described in the following section.

V. Algorithmic framework—T*ε

A. Preliminaries

Let $\mathcal{T}$ denote a set of possible transitions. By a slight abuse of notation, we assume that we have access to two functions $c : \mathcal{T} \to \mathbb{R}$ and $\hat{c} : \mathcal{T} \to \mathbb{R}$. The first, which is expensive-to-compute, corresponds to evaluating the true cost of the transition according to Eq. 4. The second, which is fast-to-compute, corresponds to evaluating a lower bound on the transition cost. Namely, $\forall T \in \mathcal{T},\ \hat{c}(T) \leq c(T)$. Once the cost of a transition T is evaluated using $c(\cdot)$, the value is stored in a cache-like data structure for later use. Thus, there is no need to compute it again when evaluating the same transition later.

To this end, let $\kappa : \mathcal{T} \to \{0, 1\}$ be an indicator function corresponding to the cases where the cost of a transition T (i) has not been evaluated using $c(\cdot)$, i.e., $\kappa(T) = 1$, or (ii) has been evaluated using $c(\cdot)$, i.e., $\kappa(T) = 0$. We start our algorithm with the setting that $\forall T \in \mathcal{T},\ \kappa(T) = 1$.[3]

B. Algorithmic Description

Our algorithmic framework, summarized in Alg. 1, consists of the following two main steps.

S1: Heuristically computing the true cost for a small set of transitions.

S2: Finding a bounded-suboptimal solution while trying to minimize the number of calls to $c(\cdot)$.

In the first step **S1** (lines 1–3), we start by finding the shortestcollision-free Dubins path $\gamma_{\text{SP}}^{v_{\min}}$ (line 1) from s_{start} to s_{goal} that uses the minimum speed $v_{\min}$. For every transition $T \in \gamma_{\text{SP}}^{v_{\min}}$ taken along this path, we compute its true cost $c(T)$, cache it, and set $\kappa(T) = 0$ (lines 2–3).

In the second step **S2** (lines 4–12), we run an A*ε-like search to find a bounded-suboptimal path from s_{start} to s_{goal} given some approximation factor ε. We base our search algorithm on A*ε because we can use it to heuristically guide the search to expand nodes for which the incoming time-optimal transition has already been computed.

Formally, each node n considered by the search is associated with a parent node n.parent except for the start node n_{start} associated with s_{start}, which has no parent. In addition, each node stores the time-optimal transition $n.T$ used to get from n.parent to n. When expanding a node n (namely, computing successor nodes and inserting them into the OPEN list), the true cost $c(n.T)$ of the time-optimal transition leading to n is computed. However, the cost used to reach a successor node n' via transition $n'.T$ is (i) $\hat{c}(n'.T)$ if $\kappa(n'.T) = 1$ and (ii) $c(n'.T)$ if $\kappa(n'.T) = 0$. Note that for some nodes in the OPEN list, the true cost of the time-optimal transition associated with them may not have been computed. However, for all expanded nodes, the true cost of the incoming time-optimal transition is computed at some point.

To this end, if we set $g(n_{\text{start}}) = 0$ to be the cost-to-come (g-value) of the start node, then the cost-to-come of any other node n in the OPEN list is

$$g(n) = g(n.\text{parent}) + \begin{cases} \hat{c}(n.\text{T}) \text{ if } \kappa(n.\text{T}) = 1, \\ c(n.\text{T}) \text{ if } \kappa(n.\text{T}) = 0. \end{cases} \tag{6}$$

The f-value of the node n (denoted by $f(n)$ and used to prioritize nodes in the OPEN list) is defined in Eq. 5. $h(n)$ can be any admissible estimate of the cost to reach the goal. In our setting, we use the length of the minimum speed Dubins path divided by the vehicle's maximum speed.

Recall (Sec. IV) that A*ε uses a FOCAL list that contains all nodes whose f-value is at most $1+\varepsilon$ times the f-value of the minimal-cost node in the OPEN list. In our setting, nodes in FOCAL are lexicographically sorted using the following key (from low to high):

$$\text{key}(n) = (\kappa(n.T), f(n)). \tag{7}$$

Thus, any node n for which $\kappa(n.T) = 0$ (i.e., the true cost $c(n.T)$ of the transition leading to n has been computed) is always prioritized before any node n' for which $\kappa(n'.T) = 1$ (i.e., the true cost $c(n'.T)$ of the transition leading to n' has not been computed), regardless of their respective f-values. Among all nodes with identical values of κ, nodes with smaller f-values are prioritized. Once a node n is chosen for expansion, the true cost of $n.T$ is computed if it has not been computed beforehand. Finally, a path is found once a node associated with the goal configuration s_{goal} is removed from the OPEN list.

Following the theoretic properties of A*ε [9] and using the fact that $\forall T\ \hat{c}(T) \leq c(T)$ we have the following Corollary.

[3]The reason we chose to define κ in this manner is that we start with $\kappa(T) = 1$ for all nodes and order nodes lexicographically in the FOCAL list where lower means better.

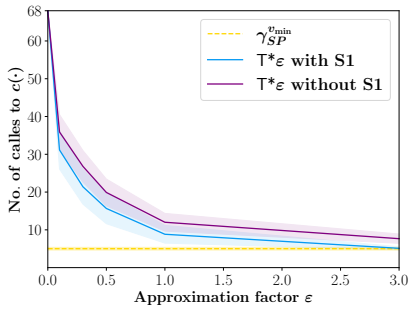


Fig. 2. Average number of calls to $c(\cdot)$ by T*ε as a function of ε with (blue) or without (purple) step **S1**, respectively. Recall that there are at most 68 possible transitions.

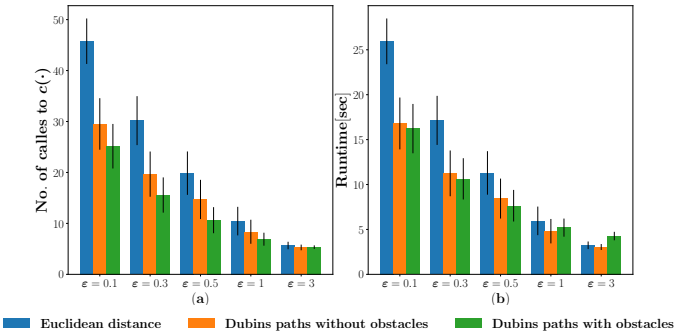


Fig. 3. Comparing the effect of different approaches for lower bounds ($\hat{c}(\cdot)$ in Eq. 6) for several values of the approximation factor ε. (a) Average number of calls to $c(\cdot)$ and (b) Average running time of T*ε, respectively.

Corollary 1: Let $\varepsilon \geq 0$ and $\hat{c}(\cdot)$ be some function that bounds from below the true cost $c(\cdot)$ of any time-optimal transition. T*ε, using $\varepsilon, \hat{c}$, and c is bounded-suboptimal with an approximation factor of $1 + \varepsilon$.

VI. Evaluation

In this section, we report on empirical evaluation of our approach in a simulated environment inspired by Song *et al.* [8]. We start (Sec. VI-A) by evaluating our heuristic approach for computing the true cost for a small set of transitions (step **S1**). Then we continue (Sec. VI-B) to demonstrate how the lower bound $\hat{c}$ needs to be both informative (i.e., as close as possible to the real cost c) and computationally cheap-to-compute (as its main purpose is to save computation time) by evaluating several different lower bounds that balance these two traits. We then move to compare our motion-planning approach with T* in static environmental conditions (Sec. VI-C), i.e., when the system dynamics are known before the query is provided and are independent of the specific location at which the robot resides, and in dynamic environmental conditions such as when wind currents exist (Sec. VI-D). Finally, we briefly discuss and qualitatively compare T*ε with alternative approaches to find paths that attempt to minimize the execution time (Sec. VI-E).

All experiments were run on a 3.1 GHz Intel Core i9 processor with 32 GB of memory. Benchmarks, system parameters, and C++ implementation of all the algorithms used in our work are publicly available[4]. Throughout Sec. VI-A–VI-C, we test the performance of our algorithm across 100 randomly generated scenarios of 14×14 cells and using $v_{\min} = 0.5$, where for a given scenario, each cell has a probability of 25 % of being blocked, and the start and goal configurations are chosen uniformly at random while ensuring that a solution exists. When reporting algorithmic attributes, e.g., solution cost or runtime, the values represent the average across the 100 scenarios. In some cases, we also report confidence intervals that include two standard deviations from the mean. In Sec. VI-D, we provide additional information on the experiment setup.

A. Evaluating the Efficacy of step *S1*

To evaluate the efficacy of the step **S1**, wherein we provide a heuristic approach for computing the true cost for a small set of transitions, we report the number of calls to $c(\cdot)$ by T*ε as a function of the approximation factor ε with and without this initial step (Fig. 2).

Observe that for values of ε larger than 0.1, using **S1** allows reducing the average number of calls to $c(\cdot)$ by roughly 10%. Moreover, in this case, for large approximation factors, the average calls to $c(\cdot)$ converges to the number of transitions in $\gamma_{SP}^{v_{\min}}$. When the step **S1** is not invoked, T*ε is not "bootstrapped" with a good set of pre-computed time-optimal and thus evaluates unnecessary transitions.

B. Computing Lower Bounds—Balancing Computation Time with Informative Lower Bounds

Recall that in Sec. V, we assumed that we have access to $\hat{c}$—an efficient-to-compute tight lower bound on c, the true cost of the time-optimal transition. However, there is an inherent tradeoff between how informative a lower bound is (which immediately corresponds to how accurately it can guide the search) and its computation time (which can negate the effectiveness of the lower bound).

We evaluated T*ε using three different lower bounds that demonstrate this behavior: (i) Euclidean distance divided by $v_{\max}$; (ii) length of minimum-speed Dubins path divided by $v_{\max}$; and (iii) length of minimum-speed Dubins path divided by $v_{\max}$ while accounting for environment obstacles. Note that all lower bounds are divided by $v_{\max}$, thus always underestimating the true cost of a transition. In addition, they are ordered from low to high according to their computation time and how informative they are.

We compare (Fig. 3) the three lower bounds w.r.t. the number of time-optimal transitions computed by T*ε and its running time. As expected, the Euclidean distance is the least-informative lower bound, thus resulting in the maximal number of time-optimal transitions computed for all values of ε. Moreover, even though it is efficient-to-compute, it does a poor job in guiding the search, and thus the runtime is high. Computing Dubins path while accounting for obstacles is more informative but also more expensive-to-compute. It does result in the smallest number of calls to $c(\cdot)$ but has longer running times when compared to using Dubins paths that do not account for obstacles. The latter also has a comparable number of calls to $c(\cdot)$. Thus, we use Dubins paths without obstacles as the lower bound in the rest of the evaluation.

[4]https://github.com/CRL-Technion/tstar-epsilon.

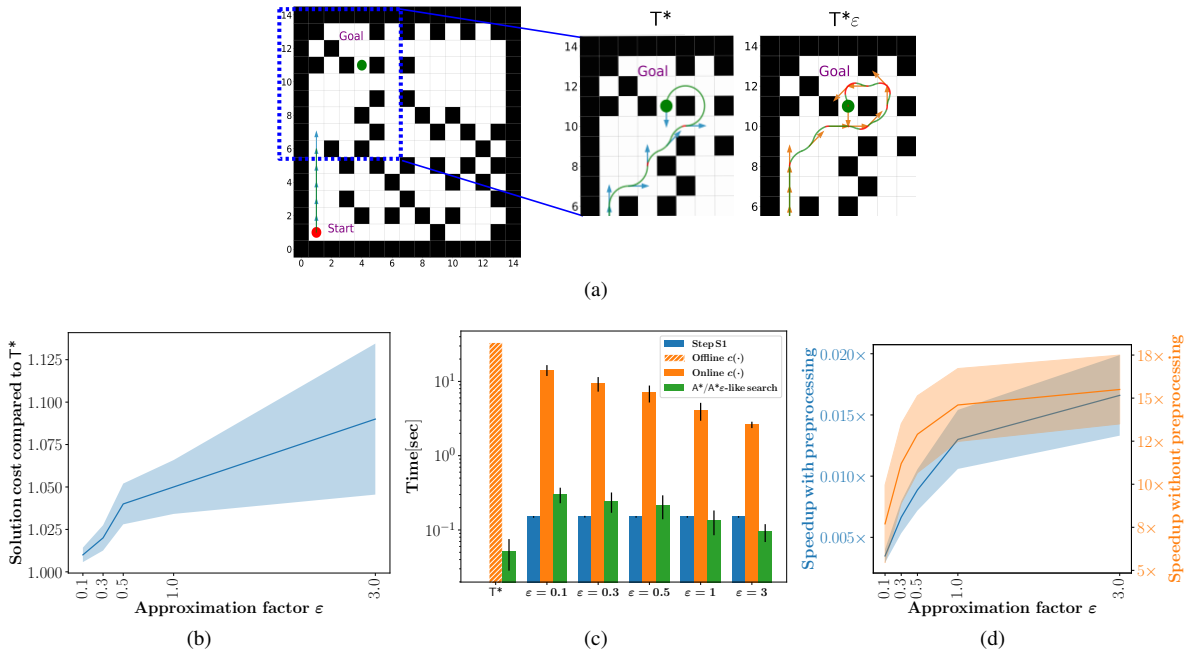


Fig. 4. Comparison of T* and T*ε in static environmental conditions over 100 randomly chosen start and goal configurations. (a) Solution found by T* and by T*ε with $\varepsilon = 3$ for one representative experiment. The beginning of the solution found by both algorithms is identical, while the end differs and is highlighted on the right-hand side. (b) Average solution cost of T*ε compared to T* as a function of ε. (c) Breakdown of the running time of T* and T*ε for different values of ε. (d) Speedup in running time of T*ε over T* obtained with and without pre-computing all time-optimal transitions.

C. Motion Planning in Static Environmental Conditions

We compare T*ε with T* in static environmental conditions, i.e., where the system dynamics are invariant to the robot's placement and orientation in the scenario. As we can see in Fig. 4c, computing time-optimal transitions dominate the running time of all algorithms. If the system dynamics are known in advance, these can be pre-computed in an offline step by T*, and it outperforms T*ε for all ε by a factor of between $60\times$ and $290\times$. However, if we account for this preprocessing time, T*ε dramatically reduce the run time by a factor ranging between $8\times$ and $15\times$. It is worth noting that in either case, T*ε finds a solution with an average cost that is much lower than the guaranteed suboptimality bound.

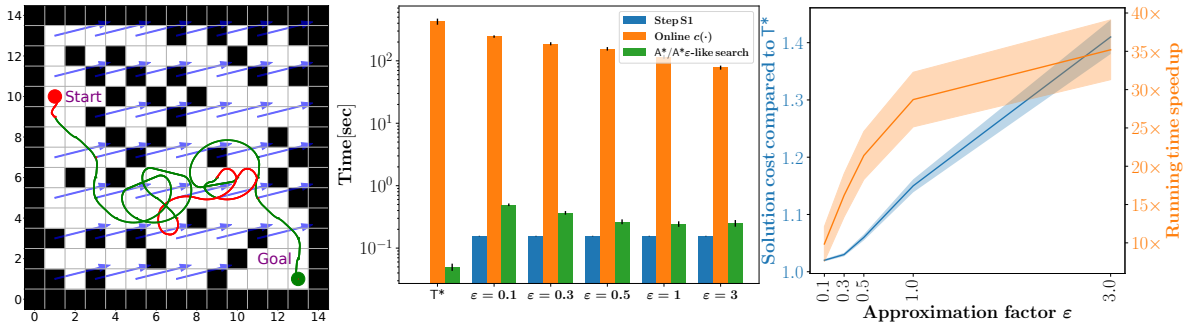
D. Motion Planning in Dynamic Environmental Conditions

In the previous section, we considered static environmental conditions and showed that using an approximation factor allows reducing T*ε's planning times dramatically. However, if the system's dynamics are known in advance, T* may perform all its computationally-expensive operations in a pre-processing stage. In contrast, this is not the case when the system's dynamics are not known in advance. It can be due to changes in the maximal speed that the system can take (due to, e.g., payload changes of the system or regulatory changes happening just before a query is received) or due to dynamic environmental conditions such as wind currents whose magnitude and direction is known only when a query is received. We use the latter case to demonstrate the efficacy of T*ε. In particular, we assume that the magnitude of wind currents is uniform across a given scenario.

It is worth noting that this dynamic setting, analyzed by Techy and Woolsey [23], can be seen as a generalization of the setting described in Sec. III and by Mittal et al. [24]. To account for this more-involved dynamic setting, we first note that one cannot use rotation and symmetry between transitions to reduce the total amount of unique transitions. Thus, in this setting, the total number of time-optimal transitions is 512. Moreover, this requires an adaptation of how time-optimal transitions are computed (see the Appendix for a description of the new model). This adaptation, based on the code for computing time-optimal transitions [18], is publicly available[5]. The second notable change is that computing lower bounds (namely, when calling $\hat{c}$, requires some care, and the modifications are detailed in the Appendix.

For dynamic environmental conditions, we used the same randomly generated scenarios as described earlier. For each such scenario, we generated 10 instances with varying wind and minimum velocity values. In particular, we sampled wind conditions from $\|\mathbf{w}\| \in [0.14, 0.41]$ and minimum velocity values $v_{\min} \in [0.4, 0.9]$. In total, we ran 1000 experiments and the averaged results appear in Fig. 5. The most computationally demanding component in both T*ε and T* is still computing the time-optimal transitions, which requires two orders of magnitude more time than any other algorithmic component. However, we observe that the approximation factor allows for a dramatic reduction of the running time with little compromise on the quality of the solution. For example, as it can be seen in Fig 5c, using $\varepsilon = 1$, T*ε guarantees a solution whose cost is at most twice $c(\gamma^*)$ the cost of the optimal solution, but returns a path whose average cost is no more than $1.15 \cdot c(\gamma^*)$. It is done while obtaining a speedup in run time by an average factor of $28\times$. In general, as depicted in Fig. 5c, increasing ε results in a dramatic running-time

[5]https://github.com/CRL-Technion/Variable-speed-Dubins-with-wind



(a) (b) (c)

Fig. 5. Comparing T* and T*ε in dynamic environmental conditions where the magnitude and direction of wind currents (blue arrows) are given at the query time. (a) Representative solution obtained on one randomly-generated map with randomly-generated start and goal configurations, wind conditions and minimum velocity $v_{\min}$. (b) Running time breakdown of T* and T*ε for the selected values of ε. (c) Average solution cost compared to the cost of the optimal solution and speedup in running time as a function of ε.

improvement at the cost of slightly lower solution quality.

E. Discussion—Alternative Approaches

Wilson *et al.* [22] reduced computational effort by allowing the system to follow Dubins paths but using several radii, and evaluated this approach when using three different radii. This heuristic approach reduces runtime, produces near-optimal solutions but with no guarantees regarding their quality. From our comparison of T*ε and Wilson's model in static environments, we found that running times are faster by several orders of magnitude, but path costs are larger by an average of $18\,\%$. However, when evaluating their approach in windy conditions, the speedup in running time is comparable to T*ε (as local transitions cannot be computed analytically and a computationally expensive root-finding problem needs to be solved [23]), and the quality of paths remain higher (again, with no formal guarantees).

An alternative approach is to compare our work with the approach by Mittal *et al.* [24] that specifically accounted for efficient computation of paths in dynamic environments. However, the main objective of their approach is online motion-planning under ocean currents, and no importance was given to the quality of paths. When comparing T*ε with their approach on representative environments, the path quality obtained by T*ε was higher (lower execution times) by a factor of more than $2.7\times$.

VII. Conclusions and Future Work

In this work, we present T*ε, a novel algorithmic framework for efficiently finding bounded-suboptimal solutions for time-optimal motion-planning. This is done by minimizing the number of computations of time-optimal transitions used by the system. When one cannot pre-compute the time-optimal transitions in advance, this reduces the planning times by orders of magnitude compared to the state-of-the-art.

Future work includes adapting our work for efficient re-planing (e.g., when the wind changes while the system is in motion or dramatic payload changes during trajectory execution changing the system's constraint). Here, we plan to adapt LPA* [28] to account for minimizing the number of transitions used.

Another natural extension is to account for settings when the approximation factor ε cannot be determined a priori by the user. Here, it is natural to use an *anytime* [29] approach. Instead of providing a fixed ε, the user would provide an upper bound for the algorithm's running time. We suggest starting with an initially high value of the approximation factor ε to compute an initial solution quickly and then progressively decreases ε as the time permits while reusing transitions computed in earlier search episodes.

Appendix

To consider the setting where we need to account for wind currents, we detail the updated dynamics model and present two approaches to compute lower bounds on transitions cost-efficiently.

A. Updated Model

We assume that there is a constant wind $\mathbf{w} = (w_x, w_y)$ that changes the original system dynamics (Eq. 1) to

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{pmatrix} = \begin{pmatrix} v(t)\cos\theta(t) + w_x \\ v(t)\sin\theta(t) + w_y \\ u(t) \end{pmatrix}. \tag{8}$$

Furthermore, we assume that regardless of the magnitude of the wind, the vehicle will move forward. It is ensured by the additional assumption that the wind magnitude is smaller than the minimum speed of the system $v_{\min}$, i.e.,

$$|\mathbf{w}| < v_{\min}. \tag{9}$$

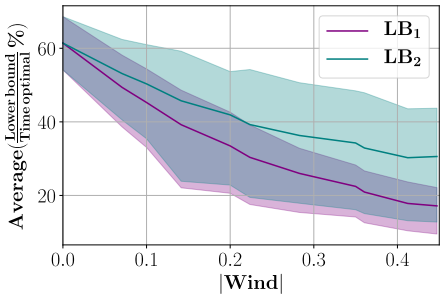


Fig. 6. Average ratio between each lower bound and the time-optimal cost as a function of the wind's modulus (solid lines). Colored regions represent a 60% non-parametric confidence interval of the true value.

B. Updated Lower Bounds

The wind component may significantly influence the turning capabilities (in the reference frame of the ground). The smallest turning radius is obtained when flying directly against the wind. We denote it by ρ_{LB}, the lower bound radius, and have that

$$\rho_{\text{LB}} = \left(1 - \frac{|\mathbf{w}|}{v_{\min}}\right) \rho_{\min}. \tag{10}$$

To this end, a lower bound on the true length of the path between two configurations s_1 and s_2 can be determined based on the length $\mathcal{L}(s_1, s_2, \rho_{\text{LB}})$ of Dubins path with the lower bound radius ρ_{LB}. Now, if this value is divided by an upper bound on the effective maximal speed $v_{\max}^{\text{effective}}$ in the reference frame of the ground, it provides a lower bound on the true cost (i.e., the travel time).

We suggest two lower bounds based on how $v_{\max}^{\text{effective}}$ is determined. The first approach assumes that $v_{\max}^{\text{effective}} = v_{\max} + |\mathbf{w}|$. Namely, the effective maximal speed is the sum of the system's maximum speed and the wind magnitude. Thus, the first lower bound LB_1 is defined as

$$\text{LB}_1(s_1, s_2) = \frac{\mathcal{L}(s_1, s_2, \rho_{\text{LB}})}{v_{\max} + |\mathbf{w}|}. \tag{11}$$

The second lower bound LB_2 considers the direction between the start and end configurations (s_1 and s_2, respectively) by computing the ground speed $\mathbf{v}_{\text{G}}$ in the given direction $\mathbf{d}$. The direction is determined based on the configurations s_1 and s_2 as $\mathbf{d} = \left(s_2^{\text{xy}} - s_1^{\text{xy}}\right) / \left(|s_2^{\text{xy}} - s_1^{\text{xy}}|\right)$, where s_i^{xy} stands for the x, y coordinates of the configuration s_i. The ground speed $\mathbf{v}_{\text{G}}$ vector can be expressed by the system's aerial speed $\mathbf{v}$ and wind $\mathbf{w}$ as $\mathbf{v}_{\text{G}} = \mathbf{v} + \mathbf{w}$.

Following Eq. 9, the maximum ground speed occurs at the system's maximum speed, i.e., $|\mathbf{v}| = v_{\max}$. The angle between $\mathbf{v}_{\text{G}}$ and $\mathbf{w}$ can be written as $\cos\angle(\mathbf{v}_{\text{G}}, \mathbf{w}) = \frac{\mathbf{d}\cdot\mathbf{w}}{|\mathbf{w}|}$. Subsequently, the ground speed can be expressed using the cosine rule as $|\mathbf{v}_{\text{G}}| = \mathbf{d}\cdot\mathbf{w} + \sqrt{(\mathbf{d}\cdot\mathbf{w})^2 + v_{\max}^2 - |\mathbf{w}|^2}$. Finally, the second lower bound LB_2 is defined as

$$\text{LB}_2(s_1, s_2) = \frac{\mathcal{L}(s_1, s_2, \rho_{\text{LB}})}{|\mathbf{v}_{\text{G}}|}. \tag{12}$$

Fig. 6 shows the average ratio between each lower bounds and the true cost as a function of the wind force. For both lower bounds, their estimation of the true cost becomes looser as the wind force increases. Notice that LB_2 is more informative than LB_1 because accounting for the direction enables reducing the upper bound on the ground speed. In addition, notice that both lower bounds reduce to the system's minimum-speed Dubins path divided by the maximum speed in static environmental conditions. Thus, we used LB_2 as lower bound for dynamic environmental conditions.

REFERENCES

[1] L. B. Kratchman, M. M. Rahman, J. R. Saunders, P. J. Swaney, and R. J. Webster III, "Toward robotic needle steering in lung biopsy: a tendon-actuated approach," in *Medical Imaging: Visualization, Image-Guided Procedures, and Modeling*, vol. 7964, 2011, p. 79641I.

[2] H. Xiang and L. Tian, "Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (UAV)," *Biosystems Engineering*, vol. 108, no. 2, pp. 174–190, 2011.

[3] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *RSS*, vol. 2, 2016, pp. 1–9.

[4] B. Garau, M. Bonet, A. Alvarez, S. Ruiz, and A. Pascual, "Path planning for autonomous underwater vehicles in realistic oceanic current fields: Application to gliders in the western mediterranean sea," *J. Marit. Res.*, vol. 6, no. 2, pp. 5–22, 2009.

[5] P. K. Agarwal, T. Biedl, S. Lazard, S. Robbins, S. Suri, and S. Whitesides, "Curvature-constrained shortest paths in a convex polygon," *SIAM J. Comput.*, vol. 31, no. 6, pp. 1814–1851, 2002.

[6] Z. Zeng, L. Lian, K. Sammut, F. He, Y. Tang, and A. Lammas, "A survey on path planning for persistent autonomy of autonomous underwater vehicles," *O.E.*, vol. 110, pp. 303–313, 2015.

[7] N. Wang, D. Zhang, L. Zhou, and Q. Liu, "Near optimal path planning for vehicle with heading and curvature constraints," in *WCICA*, 2010, pp. 4514–4519.

[8] J. Song, S. Gupta, and T. A. Wettergren, "T*: Time-optimal risk-aware motion planning for curvature-constrained vehicles," *RA-L*, vol. 4, no. 1, pp. 33–40, 2018.

[9] J. Pearl and J. H. Kim, "Studies in semi-admissible heuristics," *IEEE Trans. Pattern Anal. Mach. Intell*, no. 4, pp. 392–399, 1982.

[10] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *ICRA*, vol. 1, 2000, pp. 521–528.

[11] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *ICRA*, 2015, pp. 2951–2957.

[12] C. Dellin and S. Srinivasa, "A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors," in *ICAPS*, vol. 26, no. 1, 2016.

[13] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," *IEEE Trans. Robotics*, vol. 32, no. 3, pp. 473–483, 2016.

[14] A. Mandalika, S. Choudhury, O. Salzman, and S. Srinivasa, "Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles," in *ICAPS*, vol. 29, 2019, pp. 745–753.

[15] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: http://planning.cs.uiuc.edu/

[16] N. Haghtalab, S. Mackenzie, A. Procaccia, O. Salzman, and S. Srinivasa, "The provable virtue of laziness in motion planning," in *ICAPS*, vol. 28, no. 1, 2018.

[17] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Am. J. Math.*, vol. 79, no. 3, pp. 497–516, 1957.

[18] A. Wolek, E. M. Cliff, and C. A. Woolsey, "Time-optimal path planning for a kinematic car with variable speed," *J. Guid. Control Dyn.*, vol. 39, no. 10, pp. 2374–2390, 2016.

[19] K. Kučerová, P. Váňa, and J. Faigl, "Variable-speed traveling salesman problem for vehicles with curvature constrained trajectories," in *IROS*, 2021.

[20] J. P. Wilson, Z. Shen, S. Gupta, and T. A. Wettergren, "T*-lite: A fast time-risk optimal motion planning algorithm for multi-speed autonomous vehicles," in *OCEANS MTS/IEEE*, 2020, pp. 1–6.

[21] E. Plaku, "Robot motion planning with dynamics as hybrid search," in *AAAI*, 2013.

[22] J. P. Wilson, K. Mittal, and S. Gupta, "Novel motion models for time-optimal risk-aware motion planning for variable-speed AUVs," in *OCEANS MTS/IEEE*, 2019, pp. 1–5.

[23] L. Techy and C. A. Woolsey, "Minimum-time path planning for unmanned aerial vehicles in steady uniform winds," *J. Guid. Control Dyn.*, vol. 32, no. 6, pp. 1736–1746, 2009.

[24] K. Mittal, J. Song, S. Gupta, and T. A. Wettergren, "Rapid path planning for dubins vehicles under environmental currents," *IEEE Robot. Autom. Mag.*, vol. 134, p. 103646, 2020.

[25] J. Faigl, P. Váňa, M. Saska, T. Báča, and V. Spurnỳ, "On solution of the dubins touring problem," in *ECMR*, 2017, pp. 1–6.

[26] R. Pěnička, J. Faigl, and M. Saska, "Physical orienteering problem for unmanned aerial vehicle data collection planning in environments with obstacles," *RA-L*, vol. 4, no. 3, pp. 3005–3012, 2019.

[27] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Man Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.

[28] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artif. Intell.*, vol. 155, no. 1-2, pp. 93–146, 2004.

[29] E. A. Hansen and R. Zhou, "Anytime heuristic search," *J. Artif. Intell. Res.*, vol. 28, pp. 267–297, 2007.