Optimal Path Planning in Complex Cost Spaces With Sampling-Based Algorithms

Didier Devaurs, Thierry Siméon, and Juan Cortés

***Abstract*—Sampling-based algorithms for path planning, such as the Rapidly-exploring Random Tree (RRT), have achieved great success, thanks to their ability to efficiently solve complex high-dimensional problems. However, standard versions of these algorithms cannot guarantee optimality or even high-quality for the produced paths. In recent years, variants of these methods, such as T-RRT, have been proposed to deal with cost spaces: by taking configuration-cost functions into account during the exploration process, they can produce high-quality (i.e., low-cost) paths. Other novel variants, such as RRT*, can deal with optimal path planning: they ensure convergence toward the optimal path, with respect to a given path-quality criterion. In this paper, we propose to solve a complex problem encompassing this two paradigms: optimal path planning in a cost space. For that, we develop two efficient sampling-based approaches that combine the underlying principles of RRT* and T-RRT. These algorithms, called T-RRT* and AT-RRT, offer the same asymptotic optimality guarantees as RRT*. Results presented on several classes of problems show that they converge faster than RRT* toward the optimal path, especially when the topology of the search space is complex and/or when its dimensionality is high.**

***Note to Practitioners*—Despite their conceptual simplicity, sampling-based algorithms are very successful at solving complex, high-dimensional path-planning problems. Their underlying principle is to explore the configuration space of a mobile system by sampling it, and to build a graph representing the topology of this space. Sampling-based path planning has traditionally focused on finding *feasible* (i.e., collision-free) paths, without considering their quality. However, in many applications, it is important to compute high-quality (i.e., low-cost) paths with respect to a cost function defined on the configuration space, which is referred to as *cost-space path planning*. In recent years, variants of classical sampling-based planners have been developed to explore cost spaces. On another front, other approaches have aimed at finding the optimal (i.e., highest-quality) path with respect to a path-quality criterion, which is referred to as *optimal path planning*. In this paper, we study a problem to which little work has been devoted, and that encompasses these two paradigms: optimal path planning in a cost space. In this context, the definition of the path-quality criterion is based on the configuration-cost function. To efficiently solve this challenging problem, we propose two new sampling-based algorithms that combine the principles underlying current approaches targeting cost-space and optimal path planning. These two novel algorithms provide the same completeness and optimality guarantees as existing ones, but converge faster toward the optimal path, especially on complex planning problems. These methods have been evaluated only in simulated environments and many applications (in robotics, automation and other domains) remain to be investigated.**

***Index Terms*—Anytime path planning, cost space path planning, optimal path planning, sampling-based path planning.**

I. Introduction

ROBOT path-planning methods have traditionally focused on solving the *feasible path planning* problem, i.e., on finding a collision-free path for a robot moving in a complex environment. This relies on a classical framework abstracting the workspace of a mobile system into a *configuration space*. In many application fields, however, generating feasible solution paths might not be sufficient. It might be required to obtain a high-quality solution path with respect to a given path-quality criterion. One may even be looking for the optimal solution path with respect to this quality criterion, i.e., the path maximizing quality. This amounts to solving an *optimal path planning* problem.

The first quality criteria to be considered were path length and path duration [1]–[5]. More interesting problems can be addressed with more sophisticated criteria, based on the definition of a cost function over the configuration space, which is then referred to as a *cost space*. Early work in cost-space path planning only involved discrete, coarse-grained cost functions [2], [6]. Our work focuses on continuous configuration-cost functions, which is more challenging. As an example, in outdoor navigation problems, the cost of a configuration can be the elevation of the position of the robot within a 2-D terrain. When high-clearance paths are desirable, the cost of a configuration can be the inverse of the distance between the robot and the closest obstacle [7], [8]. Even more complex cost functions can appear in robotic problems [9], [10] and structural-biology problems [11].

When applied to the optimal path planning problem, classical grid-based methods, such as A* or D*, can compute resolution-optimal solution paths [12]. However, these methods are limited to problems involving low-dimensional spaces that can be discretized without leading to a combinatorial explosion. As an alternative, some deterministic path planners implicitly compute the optimal path with respect to a specific quality criterion. For instance, the *visibility diagram* allows obtaining the

Manuscript received November 14, 2014; revised May 13, 2015; accepted September 18, 2015. This paper was recommended for publication by Associate Editor J. Yi and Editor F. van der Stappen upon evaluation of the reviewers' comments. This work was supported by the European Community under Contract ICT 287617 "ARCAS."

D. Devaurs is with the Centre national de la recherche scientifique (CNRS), Laboratory for Analysis and Architecture of Systems (LAAS), F-31400 Toulouse, France, also with the Université de Toulouse, LAAS, F-31400 Toulouse, France, and also with the Department of Computer Science, Rice University, Houston, TX 77005 USA.

T. Siméon and J. Cortés are with Centre national de la recherche scientifique (CNRS), Laboratory for Analysis and Architecture of Systems (LAAS), F-31400 Toulouse, France, and also with the Université de Toulouse, LAAS, F-31400 Toulouse, France (e-mail: jcortes@laas.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2015.2487881

shortest path, and the *Voronoi diagram* allows generating the path with optimal clearance [13]. Nevertheless, such methods are also limited to low-dimensional spaces, and can only deal with polygonal obstacles.

On the other hand, sampling-based algorithms, such as the Rapidly-exploring Random Tree (RRT) [14], have been successful at solving complex path-planning problems in high-dimensional spaces. Besides, they are conceptually simple and achieve probabilistic completeness. Nevertheless, they originally targeted feasible path planning, and usually produce suboptimal solutions. As a complementary technique, after a solution path is computed, it is common to improve the quality of this path during a post-processing phase involving so-called "smoothing" methods [15]. However, such methods only allow to improve the path locally, and offer no guarantee of converging toward the global optimum.

With the aim of taking a configuration-cost function into account during the space exploration, a variant of RRT called Transition-based RRT (T-RRT) was proposed [7]. It extends RRT by integrating a Metropolis-like transition test. Thanks to the filtering properties of this transition test, the exploration performed by T-RRT favors low-cost regions of the space. In fact, T-RRT mostly creates new nodes in these favorable areas. T-RRT has been successfully applied to diverse robotic problems [7]–[9] and structural-biology problems [11]. Nevertheless, a drawback of T-RRT is that it cannot take a path-quality criterion into account when creating edges and, thus, involves no mechanism to allow for an improvement of the quality of the current solution path. As a consequence, it offers no optimality guarantee.

Another variant of RRT, called RRT*, was specifically devised to solve the optimal path planning problem [2]. RRT* has been shown to guarantee *asymptotic optimality*, thanks to its management of edges based on path quality. It has been applied to various robotic problems [2], [3], [16]. However, when performing optimal path planning in a cost space, RRT* is not very efficient because it does not take the configuration-cost function into account when sampling the cost space and creating new nodes. Indeed, RRT* only takes a path-quality criterion into account, when creating or removing edges. As a possible consequence, it has been observed that RRT* may converge slowly toward the optimal solution-path in high-dimensional cost-spaces [8].

In this paper, we address the issue of devising efficient algorithms that can solve difficult, optimal path-planning problems featuring complex continuous configuration-cost functions. For that, we build on the fact that RRT* and T-RRT rely on complementary concepts that can be put together. More precisely, we combine the two beneficial concepts underlying these methods: 1) the filtering properties of the transition test in T-RRT, favoring the creation of new nodes in low-cost regions of the space, and 2) the quality-based management of edges in RRT*, allowing the quality of the solution path to increase with time. We do this in two different ways, leading to two new algorithms. The first algorithm, called *Transition-based RRT** (T-RRT*), consists of integrating the transition test of T-RRT into RRT*. The motivation is to favor the exploration of low-cost regions of the space, while maintaining the asymptotic optimality property of RRT*. The second algorithm, called *Anytime T-RRT* (AT-RRT), consists of enhancing T-RRT with an anytime behavior enabled by the integration of a procedure adding useful cycles (based on the path-quality criterion) to the graph built over the space [1], [17]. The motivation is to quickly obtain a first high-quality solution path and, then, carry on the exploration for the solution to continually improve and converge toward the optimal path.

The anytime paradigm has often been applied to RRT-like algorithms. For instance, the first anytime variant of RRT was based on building successive trees and exploiting the search history [6]. The sampling process of this Anytime RRT was later improved [18]. Another anytime variant of RRT builds on the idea of pruning low-quality branches during the exploration [19]. The Rapidly-exploring Random Graph (RRG) algorithm is also an anytime variant of RRT, which consists of adding cycles to the tree built by RRT [2]. RRT* enhances RRT with an anytime behavior by rewiring the tree built by RRT [2]. Inspired by RRT*, the Rapidly-exploring RoadMaps (RRM) algorithm is another anytime RRT-like algorithm that allows balancing the exploration and refinement parts of the search process [20]. The meta-approach proposed in [21], based on short-cutting and path hybridization, can also provide RRT-like algorithms with an anytime behavior. Note that the Anytime T-RRT we introduce here differs from other anytime RRT-like algorithms in that it is based on adding *useful* cycles to the tree built by T-RRT.

In what follows, we present a simple formulation of the feasible, cost-space and optimal path planning problems (Section II). Then, we describe T-RRT* and AT-RRT in greater detail (Section III). We also explain why both algorithms are probabilistically complete and asymptotically optimal (Section IV). Finally, we evaluate T-RRT* and AT-RRT on several path planning problems, and show that they converge toward the optimal path faster than RRT* does (Section V). Thanks to the filtering properties of the transition test they include, T-RRT* and AT-RRT can efficiently solve difficult problems featuring complex cost spaces, on which RRT* converges very slowly. We present several such examples, illustrating various aspects that make a path planning problem difficult to solve. 1) If the problem features a large-scale workspace, even in low dimension, favoring low-cost regions avoids wasting time exploring the whole space. 2) If the space features several homotopic classes between which it is difficult to jump, even in low dimension, using the transition test can bias the search toward the class containing the optimal path and avoid being trapped in a suboptimal class. 3) If the problem is high-dimensional, it is inherently complex because the search space is intrinsically large and can potentially contain many homotopic classes.

This paper is an extended version of [22], in which the related work, problem formulation, and result analysis have been expanded.

II. Problem Formulation

A. Feasible Path Planning

The classical formulation of the path planning problem relies on abstracting the workspace of a robotic system into a configuration space $\mathcal{C}$, also called $\mathcal{C}$-space. A configuration $q \in \mathcal{C}$ describes the position and volume occupied by the robotic system

in the workspace. The subset of $\mathcal{C}$ containing the configurations inducing collisions with some obstacles in the workspace is denoted $\mathcal{C}_{\text{obst}}$. Assuming that its complement in $\mathcal{C}$ is an open set, we denote by $\mathcal{C}_{\text{free}}$ the set $cl(\mathcal{C} \setminus \mathcal{C}_{\text{obst}})$ of configurations producing no collision, where $cl()$ denotes the closure of a set.

Given an initial configuration $q_{\text{init}} \in \mathcal{C}_{\text{free}}$ and a goal configuration $q_{\text{goal}} \in \mathcal{C}_{\text{free}}$, a path planning problem can be defined as a triplet $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$. A path over the $\mathcal{C}$-space is a continuous function $\pi : [0,1] \rightarrow \mathcal{C}$. It is said to be collision-free if for all $t \in [0,1]$, $\pi(t) \in \mathcal{C}_{\text{free}}$, i.e., $\pi : [0,1] \rightarrow \mathcal{C}_{\text{free}}$. Let Π denote the set of all paths over $\mathcal{C}$ and Π_{free} the set of collision-free paths in Π. The *feasible path planning problem* is classically defined as follows.

Definition 1 (Feasible Path Planning Problem): Given a path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$, find a path $\pi \in \Pi_{\text{free}}$ such that $\pi(0) = q_{\text{init}}$ and $\pi(1) = q_{\text{goal}}$, if one exists, or report failure otherwise.

Let Π_{feas} denote the set of feasible paths, i.e., the set of paths in Π_{free} such that $\pi(0) = q_{\text{init}}$ and $\pi(1) = q_{\text{goal}}$. Among the path planning problems having a solution, the theoretical framework we rely on requires to focus on problems satisfying the *robust feasibility* property [2].

Based on the geometric formulation provided by the configuration space, several techniques have been proposed in the robotics community to solve the feasible path planning problem. The first ones were deterministic methods that proved to be complete: they can terminate in finite time, returning a solution if one exists, or failure otherwise. However, they cannot cope with difficult problems, and are limited to low-dimensional spaces. On the other hand, the sampling-based approaches that were proposed later on can efficiently solve high-dimensional problems. They are not complete, but satisfy a property called *probabilistic completeness*, that can be interpreted as a notion of "almost-sure" success.

Definition 2 (Probabilistic Completeness): An algorithm $\mathcal{A}$ is probabilistically complete if, for any robustly feasible path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$, the probability that $\mathcal{A}$ fails to return a solution when one exists decays to zero as the running time of $\mathcal{A}$ approaches infinity.

The analysis in Section IV is based on the fact that T-RRT and RRT* are probabilistically complete [2], [7].

B. Cost-Space Path Planning

Let $c : \mathcal{C} \rightarrow \mathbb{R}_+$ denote a continuous, differentiable cost function associating to each configuration of the $\mathcal{C}$-space a positive cost value. The *cost-space path planning problem* is denoted by a quadruplet $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c)$. Solving this problem consists of solving the feasible path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$, while taking the cost function c into account during the exploration of the $\mathcal{C}$-space. This amounts to filtering configurations based on their costs. More precisely, each method aimed at solving the cost-space path planning problem imposes a specific cost constraint evaluating each configuration, based on its cost alone, or on the cost variation associated with the local move between two configurations.

C. Optimal Path Planning

Let $c_{\text{p}} : \Pi_{\text{free}} \rightarrow \mathbb{R}_+$ denote a quality criterion, associating to each feasible path a positive cost value, and whose definition is based on the configuration-cost function $c : \mathcal{C} \rightarrow \mathbb{R}_+$. The path-quality criterion c_{p} can be defined in several ways, the most common being to consider the *integral of the cost* along a path. As a discrete approximation of the integral of the cost with constant step size $\delta = 1/n$ (where n is the number of subdivisions of the path), the cost of a path π can be defined as

$$c_{\text{p}}(\pi) = \frac{\text{length}(\pi)}{n} \sum_{k=1}^{n} c\left(\pi\left(\frac{k}{n}\right)\right). \tag{IC}$$

As an alternative, the *mechanical work* of a path can be defined as the sum of the positive cost variations along the path. This can be interpreted as summing the "forces" acting against the motion. It has been shown that the mechanical work can assess path quality better than the integral of the cost in many situations [7]. As a discrete approximation of the mechanical work with constant step size $\delta = 1/n$, the cost of a path π can be defined as

$$\sum_{k=1}^{n} \max\left\{0, c\left(\pi\left(\frac{k}{n}\right)\right) - c\left(\pi\left(\frac{k-1}{n}\right)\right)\right\}. \tag{MW}$$

We could consider other criteria to evaluate path quality, such as the maximal cost along the path, or the average cost. Which criterion is the most suited depends on the planning problem and on the characteristics of its expected optimal solution. Comparing cost criteria is out of the scope of this paper. We use both IC and MW not to limit ourselves to a single criterion, which could bias the interpretation of the results. Note that, when using such quality criteria, as a slight abuse of language, we interchangeably utilize the expressions "high-quality path" and "low-cost path."

The *optimal path planning problem* can now be defined as follows.

Definition 3 (Optimal Path Planning Problem): Given a path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$, a configuration-cost function $c : \mathcal{C} \rightarrow \mathbb{R}_+$, and a monotonic, bounded path-quality criterion $c_p : \Pi_{\text{free}} \rightarrow \mathbb{R}_+$, find a path $\pi^* \in \Pi_{\text{feas}}$ such that $c_{\text{p}}(\pi^*) = \min\{c_{\text{p}}(\pi) \mid \pi \in \Pi_{\text{feas}}\}$ if one exists, or report failure otherwise.

Based on these notations, an optimal path planning problem is denoted by a quintuplet $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_{\text{p}})$. If it admits a solution path π^*, then π^* is called the optimal path. Among the optimal path planning problems having an optimal solution path, the theoretical framework we rely on requires to focus on problems admitting a *robustly optimal* solution [2].

In the context of optimal path planning, the evaluation of a sampling-based algorithm should be based not only on the concept of probabilistic completeness, but also on the concept of *asymptotic optimality*. This property can be interpreted as a notion of "almost-sure" convergence toward the optimal path, and has been defined as follows [2].

Definition 4 (Asymptotic Optimality): An algorithm $\mathcal{A}$ is asymptotically optimal if, for any optimal path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_{\text{p}})$ admitting a robustly optimal solution path with finite cost $c^* \in \mathbb{R}_+$, the cost of the current

solution path that can be returned by $\mathcal{A}$ (this cost being infinite if no solution is available yet) decreases toward c^* as the running time of $\mathcal{A}$ approaches infinity.

The analysis in Section IV is based on the asymptotic optimality of RRT* [2].

III. Algorithms

The Rapidly-exploring Random Tree (RRT) [14] is a popular sampling-based algorithm that can solve the feasible path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$. Starting from the initial configuration q_{init}, RRT iteratively builds a tree $\mathcal{T}$ on the $\mathcal{C}$-space. At each iteration, a configuration q_{rand} is randomly sampled in $\mathcal{C}$, and an extension toward q_{rand} is attempted, starting from its nearest neighbor, q_{near}, in $\mathcal{T}$. If the extension succeeds, a new configuration q_{new} is added to $\mathcal{T}$, and connected by an edge to q_{near}. The criteria on when to stop can be reaching the goal configuration q_{goal}, a given number of nodes in $\mathcal{T}$, a given number of iterations, or a given running time.

Several algorithms have been devised as extensions of RRT to explore cost spaces. Among them, the T-RRT consists of integrating in RRT a transition test that favors the exploration of low-cost regions of $\mathcal{C}$ [7]. This transition test is used to accept or reject the move from q_{near} to q_{new} based on their respective costs. Even though it yields high-quality (i.e., low-cost) paths when solving the cost-space path planning problem, T-RRT offers no guarantee to solve the optimal path planning problem because it does not include any mechanism to improve its solution.

The other variant of RRT we consider here, named RRT*, has been specifically developed to solve the optimal path planning problem [2]. In RRT*, instead of being linked to q_{near}, q_{new} is linked to the configuration (among its neighbors in $\mathcal{C}$) maximizing the quality of the path in $\mathcal{T}$ between q_{init} and q_{new}. Furthermore, if, as a parent in $\mathcal{T}$, q_{new} allows one of its neighbors in $\mathcal{C}$ to be connected to q_{init} via a higher-quality path than the one currently available, some rewiring is performed in $\mathcal{T}$. By deciding how to create and remove edges of $\mathcal{T}$ based on the quality of the paths between q_{init} and every node in $\mathcal{T}$, RRT* enables the quality of the solution extracted from $\mathcal{T}$ to increase with time. However, despite its asymptotic-optimality guarantees, RRT* may converge slowly in high-dimensional cost spaces [8].

In this work, we combine the beneficial (and complementary) concepts underlying RRT* and T-RRT. These concepts are: 1) the filtering properties of the transition test in T-RRT, favoring the creation of new nodes in low-cost regions of the space and 2) the quality-based management of edges in RRT*, allowing the quality of the solution path to increase with time. We do this in two different ways, by proposing an extension to RRT* named *Transition-based* RRT* (T-RRT*) and an extension to T-RRT named *Anytime* T-RRT (AT-RRT). Both algorithms offer asymptotic-optimality guarantees when applied to the optimal path planning problem. They allow one to efficiently explore complex continuous cost-spaces, yielding high-quality solution paths that improve with time in an anytime fashion.

A. Transition-Based RRT* (T-RRT*)

The pseudo-code of T-RRT* is presented in Algorithm 1. It extends RRT* by integrating the transition test (line 6) originally developed for T-RRT [7]. This transition test is used to accept or reject the move from q_{near} to q_{new} based on their respective costs. If the move is accepted, T-RRT* behaves exactly like RRT*. First, a new node is created in $\mathcal{G}$ to store q_{new} (line 7). Then, a search in $\mathcal{G}$ is performed to compute the set Q_{near} of configurations contained in a neighborhood of q_{new} of radius $\gamma(\log(n)/n)^{1/d}$ (line 9). As defined for RRT*, this radius depends on the dimension d of $\mathcal{C}$, on a constant γ derived from the volume of $\mathcal{C}_{\text{free}}$, and on the number n of nodes in $\mathcal{G}$ [2]. This dependency on n ensures that the radius decreases as $\mathcal{G}$ grows. The next step of the algorithm consists of finding the configuration q_{par} in $Q_{\text{near}} \cup \{q_{\text{near}}\}$ to which q_{new} should be connected (line 10): the parent of q_{new} is chosen as the configuration via which the path between q_{init} and q_{new} has minimal cost. This is done by computing, for all $q_{\text{n}} \in Q_{\text{near}} \cup \{q_{\text{near}}\}$, the cost $c_{\text{p}}(\pi^{\mathcal{G}}_{\text{n}}) + c_{\text{p}}(\pi^{\mathcal{C}}_{\text{n}})$, where $\pi^{\mathcal{G}}_{\text{n}}$ is the path between q_{init} and q_{n} in $\mathcal{G}$, and $\pi^{\mathcal{C}}_{\text{n}}$ is the path between q_{n} and q_{new} in $\mathcal{C}$. Finally, since the addition of a new node in $\mathcal{G}$ potentially leads to the apparition of new paths having lower costs than those currently in $\mathcal{G}$, some rewiring might be performed (lines 12–16). For each $q_{\text{n}} \in Q_{\text{near}}$, if the cost of the path going from q_{init} to q_{n} via q_{new} is lower than the cost of the current path between q_{init} and q_{n} in $\mathcal{G}$, q_{new} becomes the new parent of q_{n} in $\mathcal{G}$.

Algorithm 1: Transition-based RRT* (T-RRT*)

input : the optimal path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_{\text{p}})$
the dimension d of the $\mathcal{C}$-space
the γ constant derived from the volume of $\mathcal{C}_{\text{free}}$ [2]
output: the graph $\mathcal{G}$

1 $\mathcal{G} \leftarrow$ `initGraph`(q_{init})
2 **while not** `stoppingCriteria`$(\mathcal{G})$ **do**
3 $q_{\text{rand}} \leftarrow$ `sampleRandomConfiguration`$(\mathcal{C})$
4 $q_{\text{near}} \leftarrow$ `findNearestNeighbor`$(\mathcal{G}, q_{\text{rand}})$
5 $q_{\text{new}} \leftarrow$ `extend`$(q_{\text{near}}, q_{\text{rand}})$
6 **if** $q_{\text{new}} \neq$ null **and** `transitionTest`$(\mathcal{G}, c(q_{\text{near}}), c(q_{\text{new}}))$ **then**
7 `addNewNode`$(\mathcal{G}, q_{\text{new}})$
8 $n \leftarrow$ `numberOfNodes`$(\mathcal{G})$
9 $Q_{\text{near}} \leftarrow$ `nodesInBall`$(\mathcal{G}, q_{\text{new}}, \gamma\,(\log(n)\,/\,n)^{1\,/\,d})$
10 $q_{\text{par}} \leftarrow$ `node_minCost`$_{\text{init}}(q_{\text{new}}, q_{\text{near}}, Q_{\text{near}}, c_{\text{p}})$
11 `addNewEdge`$(\mathcal{G}, q_{\text{par}}, q_{\text{new}})$
12 **foreach** $q_{\text{n}} \in Q_{\text{near}}$ **do**
13 $\pi \leftarrow$ `pathInSpace`$(q_{\text{new}}, q_{\text{n}})$
14 **if** `cost`$_{\text{init}}(q_{\text{new}}) + c_{\text{p}}(\pi) <$ `cost`$_{\text{init}}(q_{\text{n}})$ **and** `isCollisionFree`(π) **then**
15 `removeEdge`$(\mathcal{G}, \text{parent}(q_{\text{n}}), q_{\text{n}})$
16 `addNewEdge`$(\mathcal{G}, q_{\text{new}}, q_{\text{n}})$
17 **return** $\mathcal{G}$

The `transitionTest` involved in the T-RRT* algorithm is presented in Algorithm 2. The transition between two configurations is evaluated on the basis of their costs c_{i} and c_{j}, c_{i} being the cost of the source configuration and c_{j} the cost of the target configuration. A downhill move $(c_{\text{j}} \leq c_{\text{i}})$ in the cost landscape is always accepted. An uphill move is accepted or rejected based on the probability $\exp(-(c_{\text{j}} - c_{\text{i}})/T)$ that decreases exponentially with the cost variation $c_{\text{j}} - c_{\text{i}}$. In that case, the level of selectivity of the transition test is controlled by the *temperature* T, which is an adaptive parameter of the algorithm. Low temperatures limit the expansion to gentle slopes of the cost landscape, and high temperatures enable it to climb steep slopes. After each accepted uphill move, T is decreased to avoid over-exploring

Algorithm 2: transitionTest ($\mathcal{G}$, c_i, c_j)

input : the current temperature T
the temperature increase rate T_rate
output: *true* if the transition is accepted, *false* otherwise
1 **if** $c_\mathrm{j} \leq c_\mathrm{i}$ **then return** True
2 **if** $\exp(-(c_\mathrm{j} - c_\mathrm{i}) \,/\, T) > 0.5$ **then**
3 $T \leftarrow T \,/\, 2^{(c_\mathrm{j}-c_\mathrm{i}) \,/\, \texttt{costRange}(\mathcal{G})}$; **return** True
4 **else**
5 $T \leftarrow T \cdot 2^{T_\mathrm{rate}}$; **return** False

Algorithm 3: Anytime Transition-based RRT (AT-RRT)

input : the optimal path planning problem $(\mathcal{C}, q_\mathrm{init}, q_\mathrm{goal}, c, c_\mathrm{p})$
output: the graph $\mathcal{G}$
1 $\mathcal{G} \leftarrow$ initGraph(q_init)
2 **while not** stoppingCriteria($\mathcal{G}$) **do**
3 $q_\mathrm{rand} \leftarrow$ sampleRandomConfiguration($\mathcal{C}$)
4 $q_\mathrm{near} \leftarrow$ findNearestNeighbor($\mathcal{G}$, q_rand)
5 $q_\mathrm{new} \leftarrow$ extend(q_near, q_rand)
6 **if** $q_\mathrm{new} \neq$ null **and**
transitionTest($\mathcal{G}$, $c(q_\mathrm{near})$, $c(q_\mathrm{new})$) **then**
7 addNewNode($\mathcal{G}$, q_new)
8 addNewEdge($\mathcal{G}$, q_near, q_new)
9 **if** solutionPathExists($\mathcal{G}$, q_init, q_goal) **then**
10 addUsefulCycles($\mathcal{G}$, q_new, c_p)
11 **return** $\mathcal{G}$

high-cost regions: it is divided by $2^{(c_\mathrm{j}-c_\mathrm{i})/\texttt{costRange}(\mathcal{G})}$, where $\texttt{costRange}(\mathcal{G})$ is the cost difference between the highest-cost and the lowest-cost configurations stored in the nodes of $\mathcal{G}$. After each rejected uphill move, T is increased to facilitate the exploration and avoid being trapped in a local minimum of the cost landscape: it is multiplied by 2^{T_rate}, where $T_\mathrm{rate} \in (0, 1]$ is the increase rate of the temperature.

B. Anytime Transition-Based RRT (AT-RRT)

The pseudo-code of AT-RRT is shown in Algorithm 3. It also features the `transitionTest` (line 6) shown in Algorithm 2, and extends T-RRT by offering an anytime behavior. Before any solution is found, AT-RRT behaves exactly like T-RRT (lines 3–8). As opposed to what happens in T-RRT, however, after a solution path is found, the exploration is allowed to continue and a cycle-addition procedure is activated (lines 9–10). This procedure is based on the notion of *useful cycles*, as described in [1] and [17]. It leads to the creation in $\mathcal{G}$ of new paths that can be of higher quality than the best one found so far. This allows the current solution-path to be continually improved, in an anytime fashion, and to converge toward the optimal path.

The `addUsefulCycles` procedure is presented in Algorithm 4. When a new configuration q_new is added to $\mathcal{G}$, we consider all other configurations in $\mathcal{G}$, within a neighborhood of q_new, as potential candidate targets for new edges. The radius of this neighborhood depends on the dimension d of the $\mathcal{C}$-space and on a constant γ derived from the volume of $\mathcal{C}_\mathrm{free}$, as is done for RRT* [2]. Again, this radius decreases with the number n of nodes in $\mathcal{G}$. Within the candidate set Q_near, we are interested in the configurations that are "close" to q_new in $\mathcal{C}$, but "far" from q_new in $\mathcal{G}$, not in terms of distance but in terms of path cost. For each candidate $q_\mathrm{n} \in Q_\mathrm{near}$, if the cost of the local path π_s between q_new and q_n in $\mathcal{C}$ is strictly less than the cost of the lowest-cost path π_g between q_new and q_n in $\mathcal{G}$, and

Algorithm 4: addUsefulCycles ($\mathcal{G}$, q_new, c_p)

input: the dimension d of the $\mathcal{C}$-space
the γ constant derived from the volume of $\mathcal{C}_\mathrm{free}$ [2]
1 $n \leftarrow$ numberOfNodes($\mathcal{G}$)
2 $Q_\mathrm{near} \leftarrow$ nodesInBall($\mathcal{G}$, q_new, $\gamma\,(\log(n) \,/\, n)^{1\,/\,d}$)
3 **foreach** $q_\mathrm{n} \in Q_\mathrm{near}$ **do**
4 $\pi_\mathrm{g} \leftarrow$ pathInGraph($\mathcal{G}$, q_new, q_n)
5 $\pi_\mathrm{s} \leftarrow$ pathInSpace(q_new, q_n)
6 **if** $c_\mathrm{p}(\pi_\mathrm{s}) < c_\mathrm{p}(\pi_\mathrm{g})$ **and** isCollisionFree(π_s) **then**
7 addNewEdge($\mathcal{G}$, q_new, q_n)

if π_s is collision-free, we add an edge to $\mathcal{G}$ between q_new and q_n, thus creating a useful cycle.

IV. Theoretical Arguments

We now review the properties of the T-RRT* and AT-RRT algorithms, in terms of probabilistic completeness and asymptotic optimality (cf. Section II). It has already been proven in previous work that the T-RRT and RRT* algorithms are probabilistically complete [2], [7]. In the case of T-RRT, this property is directly derived from the probabilistic completeness of RRT: despite the integration of the transition test in RRT to devise T-RRT, the algorithm retains this property. A similar reasoning allows us to state that T-RRT* is probabilistically complete, thanks to the probabilistic completeness of RRT*. Furthermore, as AT-RRT behaves like T-RRT before a solution path is found, it satisfies the same properties.

Theorem 1 (Probabilistic Completeness): The T-RRT* and AT-RRT algorithms are probabilistically complete.

Let us assume that the γ constant involved in T-RRT* and AT-RRT, and originally introduced in RRT*, satisfies

$$\gamma > 2 \left(1 + \frac{1}{d}\right)^{\frac{1}{d}} \left(\frac{\mu(\mathcal{C}_\mathrm{free})}{\zeta_d}\right)^{\frac{1}{d}} \tag{1}$$

where d is the dimension of the $\mathcal{C}$-space, ζ_d is the volume of the unit ball in the d-dimensional Euclidean space, and $\mu()$ is an operator measuring the volume of a space. Under this assumption, it has been proven that RRT* is asymptotically optimal [2].

The only difference between T-RRT* and RRT* is the addition of the transition test, that filters the nodes added to $\mathcal{G}$ based on their costs. However, applying such filtering has no impact on the current analysis. First, the probability of any sampled node to be accepted by the transition test is never zero: this probability depends exponentially on the local cost increase and on the value of the temperature parameter. Second, even if a sample q_new is rejected at some point in time, the probability of accepting another sample infinitely close to q_new later on tends to one when running time tends to infinity. Therefore, even though T-RRT* favors the sampling of low-cost areas of $\mathcal{C}_\mathrm{free}$ at the beginning of the exploration, if it were to run for an infinite amount of time, T-RRT* would generate a set of nodes uniformly covering $\mathcal{C}_\mathrm{free}$; no region in $\mathcal{C}_\mathrm{free}$ is prevented from being explored. In other words, the nodes generated by T-RRT* can be assumed to be drawn from a uniform distribution, as is the case for RRT*. Based on the proof provided for RRT* [2], we can thus argue that T-RRT* is also asymptotically optimal.

The lower bound on γ expressed in (1) is the minimal value allowing RRT* to be asymptotically optimal. Because increasing the value of γ raises the computational cost of an

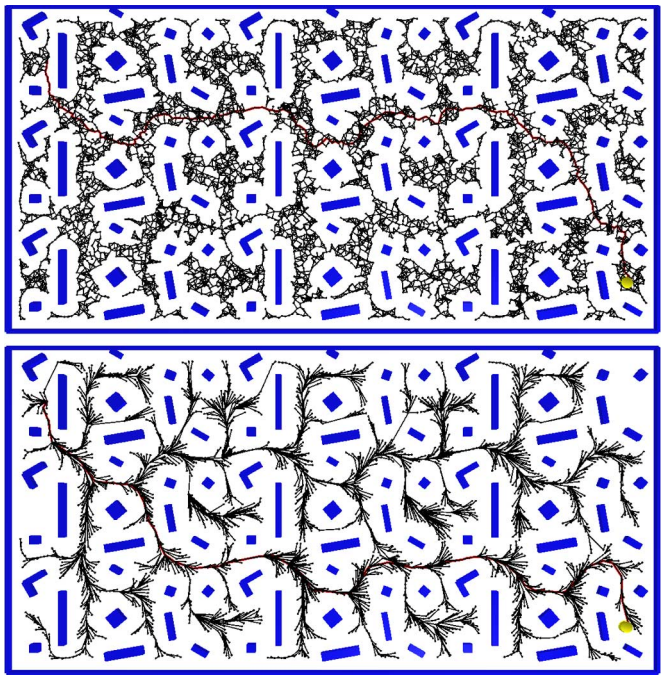

Fig. 1. *Stones* problem: 2-DoF disk moving among rectangular-shaped obstacles, while having to maximize its clearance. The figure shows graphs built by AT-RRT (top) and T-RRT* (bottom) after a running time of 0.5 s.

iteration of RRT* (due to the increased number of neighbors to be considered), this lower bound and the associated connection radius represent the optimal tradeoff between efficiency and asymptotic optimality. The connection radius involved in RRT* is based on the number of nodes in the graph at a given point in time, and not on the number of expansion attempts. The way nodes are generated has no influence on the way the connection radius is defined, as long as these nodes are drawn from a uniform distribution. Therefore, the optimal connection radius is the same for RRT* and T-RRT*.

AT-RRT and T-RRT* use the same procedure to create and filter nodes, based on the extension mechanism of RRT and on the transition test of T-RRT. The difference between them lies only in the management of edges, and has no impact on the current analysis. Both algorithms make the decisions to create alternative paths based on cost improvement. The specific criterion that an edge has to satisfy to be considered useful in terms of cost improvement does not feature in the proof of asymptotic optimality of RRT*. As the point processes associated with AT-RRT and T-RRT* are the same, AT-RRT is also asymptotically optimal.

Theorem 2 (Asymptotic Optimality): The T-RRT* and AT-RRT algorithms are asymptotically optimal.

V. Evaluation

A. Path Planning Problems

We have evaluated the T-RRT* and AT-RRT algorithms on several optimal path-planning problems that differ in terms of $\mathcal{C}$-space dimensionality, geometrical complexity and configuration-cost function type.

The *Stones* problem (illustrated in Figs. 1, 5, and 9) is a 2-degrees-of-freedom (DoFs) example in which a disk has to go through a space cluttered with rectangular-shaped obstacles. The objective is to maximize clearance, so the cost function associates to each position of the disk the inverse of the distance between the disk and its closest obstacle.

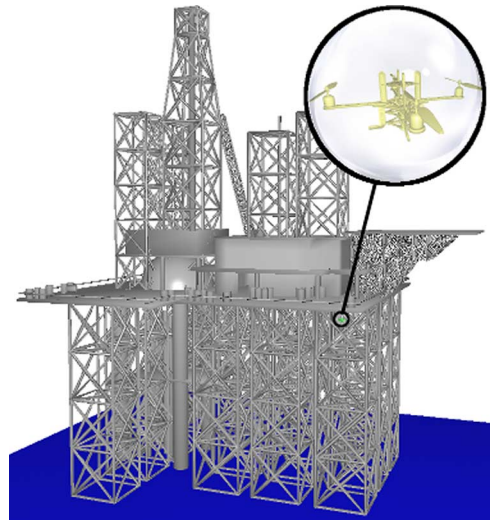

Fig. 2. *Inspection* problem: quadrotor (whose close-up is shown in yellow) inspecting an oil platform. The cost function involves the clearance of the quadrotor's 3-DoF safety sphere.

The *Inspection* problem deals with industrial inspection in a dense environment. It involves an aerial robot used to inspect an oil platform, as shown in Fig. 2. The featured quadrotor is modeled as a 3-DoF sphere (i.e., a free-flying sphere) representing the security zone around it. Assuming that motions are performed quasi-statically, we restrict the problem to planning in position (thus disregarding control issues). For safety reasons, the quadrotor has to move in this environment trying to maximize clearance for the security sphere. The specificity of this problem is its large-scale workspace.

The *Transport* problem features aerial robots, and deals with the collaborative transport of objects, as shown in Fig. 3.Two quadrotors have to carry an H-looking object and go through one of two holes in a wall. The robotic system comprises the quadrotors themselves (and not safety spheres around them), the 3-R planar manipulator arms attached below them, and the carried object. A configuration of this system is defined by the position and orientation of the object in space, and the relative positions of the quadrotors with respect to the object. This problem is restricted to planning in position for the quadrotors because of the quasi-static assumption made on their motions. We consider a planar version of the problem, thus disregarding translations along the Y axis and rotations around the X and Z axes. Besides, the revolute joints of the arms are dependent degrees-of-freedom in constraints related to the closure of the kinematic chain. Therefore, the system can be described with 7 DoFs: 3 DoFs for the object (two translations along the X and Z axes, and a rotation around the Y axis) and 2 DoFs for each quadrotor (two translations along the X and Z axes). In this example, different cost functions can be defined. The notion of clearance could be considered, but we use a cost function based on the notion of "balance" in our experiments. Assuming the initial configuration is stable, the idea is to maintain it as much as possible, while allowing a complete freedom of movement

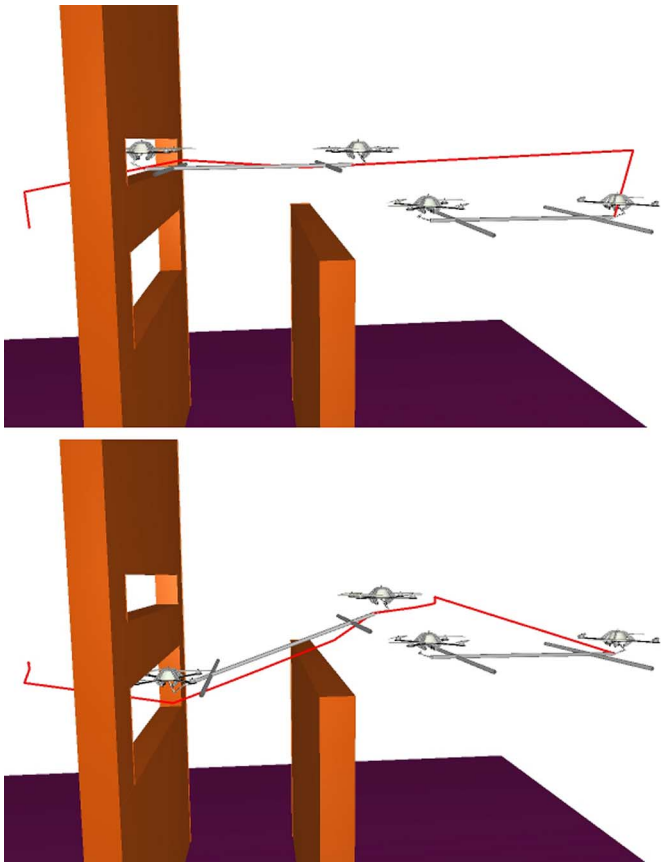

Fig. 3. *Transport* problem: two quadrotors have to transport an object and go through one of the holes in the wall, while maintaining the balance of the whole system. Both images show an intermediate configuration as well as the goal configuration along paths obtained after 50 s. Top: path produced by T-RRT* when minimizing MW. Paths obtained when minimizing IC, and paths produced by AT-RRT are similar. Bottom: path produced by RRT* when minimizing IC. Paths obtained when minimizing MW are similar.

for the object with respect to the translations along the X and Z axes. To achieve that, the cost of a configuration is defined as the sum of the differences to the initial values for the rotation of the object and the translations of the quadrotors. The specificity of the *Transport* problem lies in the fact that it features two very distinct homotopic classes. The two holes in the wall constitute narrow passages of similar difficulty in terms of purely geometrical planning: despite being wider, the lower hole is partly obstructed by the second wall. Therefore, a geometrical planner such as RRT produces feasible paths going through either hole with similar probability. However, when planning in the cost space with the clearance-based cost function, paths going through the lower hole are favored because it is larger than the other one. On the contrary, when planning in the cost space with the balance-based cost function, paths going through the upper hole are favored because they do not require the robotic system to tilt sharply.

The *Snake* problem (illustrated in Fig. 4) involves a snake-like object constituted of ten identical cylinders between which nine revolute joints are defined. We also consider two translations and a rotation of the whole system in the planar workspace, which adds up to 12 DoFs. The cost function is defined as the sum of the absolute differences between the angular values of

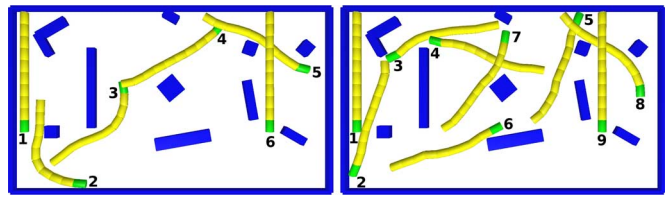

Fig. 4. Selected configurations along paths produced by AT-RRT when minimizing IC (left) or MW (right), after a running time of 100 s, on the *Snake* problem. A snake-like object has to move among rectangular obstacles. The cost function favors straight configurations, and regular over irregular coiling. T-RRT* provides similar results.

consecutive revolute joints, added to the absolute value of the first revolute joint. It favors a straight configuration of the object, or configurations in which all revolute joints have the same value, which correspond to a regular coiling of the object. This problem enables us to analyze the behavior of the algorithms in higher dimension.

B. Settings

Before applying T-RRT* and AT-RRT to these path planning problems, the values of their parameters have to be set. Following [8], T_{rate} is set to 0.1 and T is initialized to 10^{-6}. The choice of these values is based on the analysis performed in [7]. Finding a good value for γ happens to be a real issue. As already mentioned, the lower bound for γ expressed in (1) is the optimal value with respect to the tradeoff between efficiency and asymptotic optimality. However, computing this value requires to estimate the volume of $\mathcal{C}_{\text{free}}$. This is possible in low-dimensional spaces when the robotic system and the obstacles are represented with simple geometric models, but this is not realistic otherwise. To ensure that γ satisfies (1), we use the value

$$\gamma = 2\left(1+\frac{1}{d}\right)^{\frac{1}{d}}\left(\frac{\mu(\mathcal{C})}{\zeta_d}\right)^{\frac{1}{d}}. \qquad (2)$$

On the *Stones* and *Inspection* problems, since $\mathcal{C}$ is an Euclidean space, its volume $\mu(\mathcal{C})$ can be computed using the validity interval of each DoF. However, this is not straightforward on the *Transport* and *Snake* problems because of the revolute joints they involve. For each DoF corresponding to such joint, its angular range is multiplied by the length of the associated rigid body.

T-RRT* and AT-RRT have been implemented in the path-planning platform *Move3D* [23]. To fairly assess them, no smoothing is performed on the solution paths. Values of IC (integral of cost) and MW (mechanical work) are averaged over 100 runs. Results have been obtained on an Intel Core i5 processor at 2.6 GHz with 8 GB of memory.

C. Results

T-RRT* and AT-RRT build graphs over $\mathcal{C}$ in different ways because they involve different strategies to create (and potentially remove) edges. This is illustrated in Fig. 1 on the *Stones* problem. The upper figure clearly shows the cycles created by AT-RRT, and the redundancy in paths. As can been seen in the lower figure, the tree built by T-RRT* is much sparser, because high-cost edges are removed. The numerical results we present show that these differences in behavior do not create significant

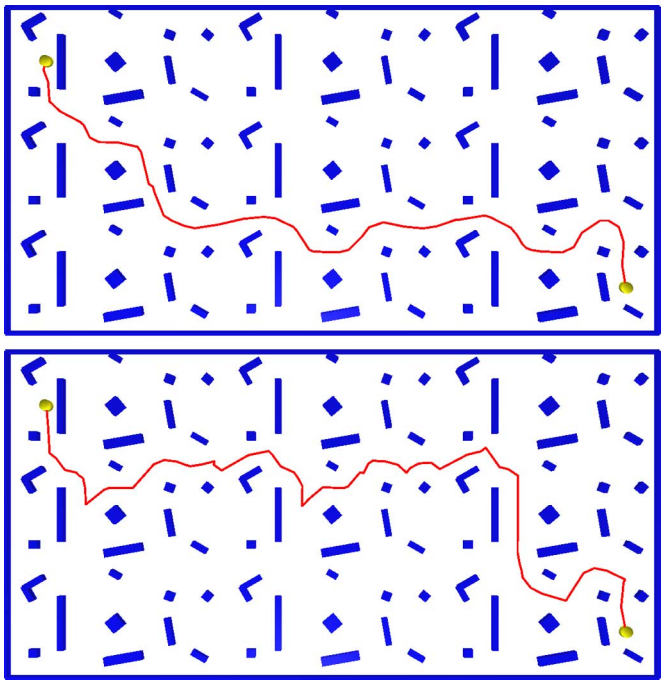

Fig. 5. Solution paths produced by T-RRT* on the *Stones* problem when minimizing IC (top) or MW (bottom) after a running time of 10 s. Solution paths produced by AT-RRT are similar.

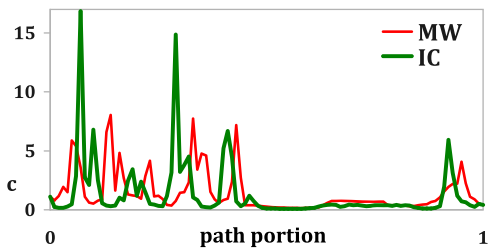

Fig. 6. Cost profiles of two paths produced by AT-RRT on the *Inspection* problem when minimizing IC or MW, after a running time of 10 sec.

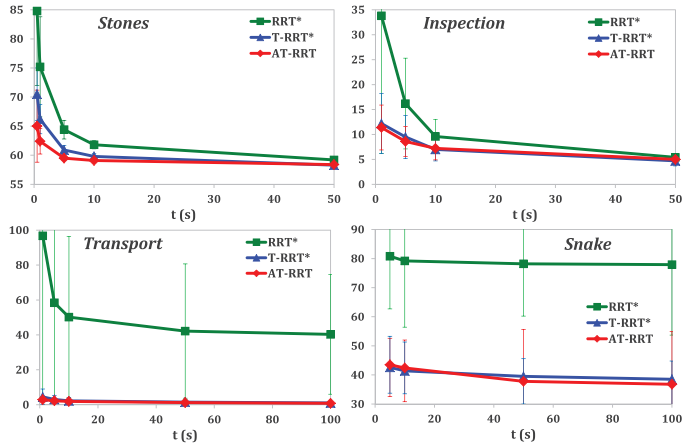

Fig. 7. Evolution over time of the costs (integral of the cost, IC, averaged over 100 runs) of the solution paths produced by RRT*, T-RRT*, and AT-RRT, on the four path planning problems.

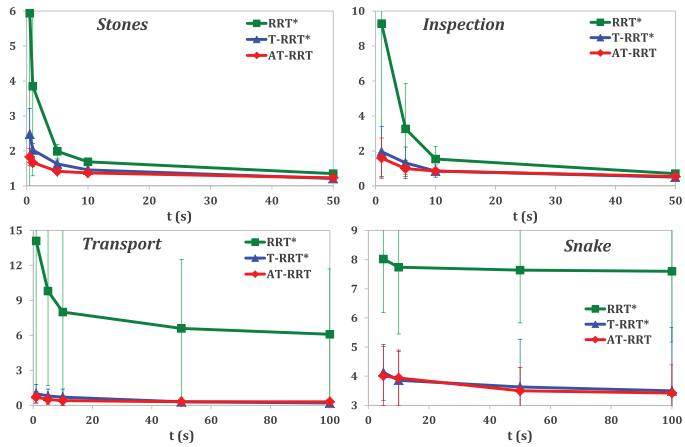

Fig. 8. Evolution over time of the costs (mechanical work, MW, averaged over 100 runs) of the solution paths produced by RRT*, T-RRT*, and AT-RRT, on the four path planning problems.

differences in performance. Also, the solution paths produced by the two algorithms usually look very similar.

Differences in solution paths are mainly due to the choice of the path-quality criterion: IC or MW. This is clearly visible in Figs. 5 and 6. Minimizing IC tends to favor shorter paths along which the maximal cost can be very high (as illustrated by Fig. 6), and minimizing MW sometimes produces strangely convoluted paths (as illustrated by the lower part of Fig. 5). Another drawback of MW (not illustrated here) is that, if the cost of q_{init} is high, MW can be low even for paths going through high-cost configurations. A better cost criterion could probably be defined by combining the expressions of IC and MW, but this goes beyond the scope of this paper. Note that, on some problems, such as *Transport*, the choice of the cost criterion has little impact on the results.

To evaluate the performance of T-RRT* and AT-RRT, we analyze the evolution over time of the costs of the solution paths they produce. As a reference, we compare both algorithms to RRT* [2]. To obtain the best results with RRT*, we use the *conditional activation* and *branch-and-bound* heuristics when they are beneficial. The *conditional activation* heuristic consists of planning with a regular RRT until the first solution is found, and only then activating the procedures specific to RRT* [16]. The *branch-and-bound* heuristic consists of trimming the nodes in $\mathcal{G}$ that cannot allow finding paths with costs lower than that of the current solution path, which is assessed using a cost-to-go function [3]. Both heuristics are beneficial on the *Transport* and *Snake* problems.

Numerical results obtained on the four path planning problems (each one being tested with a given pair (q_{init}, q_{goal}) of configurations) are reported in Fig. 7 for IC, and Fig. 8 for MW. They clearly show that T-RRT* and AT-RRT converge faster than RRT* toward the optimum. Even on a problem as simple as *Stones*, if only little time is available, T-RRT* and AT-RRT yield better-quality solutions than RRT*. But, given enough time, all algorithms produce paths of similar quality. Analyzing the exploratory behavior of each algorithm reveals that the filtering properties of the transition test help focus the search on the most relevant parts (i.e., the low-cost areas) of the workspace. Indeed, graphs produced by RRT* contain numerous nodes in high-cost regions of the space (as is the case

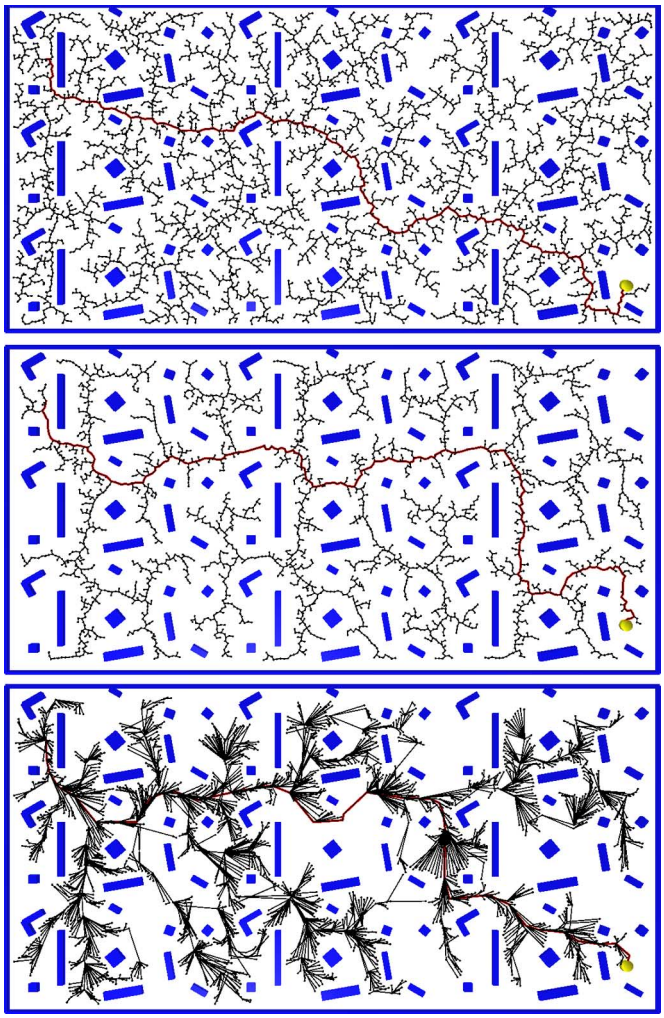

Fig. 9. Graphs built on the *Stones* problem, by RRT (top), T-RRT (middle), and RRT* (bottom). As this example involves a clearance-based cost function, nodes close to the obstacles have high costs, and nodes far from the obstacles have low costs. Because they do not take this configuration-cost function into account, RRT and RRT* create nodes close to the obstacles. On the contrary, as it favors low-cost regions of the space, T-RRT creates nodes further from the obstacles, similarly to what T-RRT* and AT-RRT do (cf. Fig. 1).

with RRT, cf. Fig. 9), contrary to graphs produced by T-RRT* or AT-RRT (cf. Fig. 1).

When the size of the workspace is larger, as in the *Inspection* problem, the dominance of T-RRT* and AT-RRT is even clearer. When the problem is even more complex, as is the case of *Transport*, the weaknesses of RRT* start to translate into a very low rate of convergence. Thanks to the transition test, the search performed by T-RRT* or AT-RRT is usually guided toward the homotopic class containing the optimal path (i.e., the upper hole, when using the balance-based cost function, as shown in the upper part of Fig. 3). On the contrary, the first solution produced by RRT* can belong to any of the two homotopic classes; if it is found in the suboptimal one (i.e., the lower hole), RRT* gets stuck in this class and into optimizing a low-quality solution (see the lower part of Fig. 3).

On high-dimensional problems, such as *Snake*, RRT* converges very slowly. Looking at Figs. 7 and 8, one may think that it is also the case for T-RRT* and AT-RRT. To check that, we have let the three algorithms run for 12 hours, while minimizing MW. We have obtained solutions of costs 3.42, 2.41, and 2.24 for RRT*, T-RRT*, and AT-RRT, respectively. Looking at Fig. 8, it means that, after 100 s, T-RRT* and AT-RRT are already close to the optimum, contrary to RRT*.

Finally, to assess whether what we observe is consistent across the domains corresponding to the four optimal path-planning problems, we have evaluated the algorithms on instances of these problems involving different pairs $(q_{\text{init}}, q_{\text{goal}})$ of configurations. The results we have obtained (but that are not presented here due to space limitations) are similar to what we report above.

VI. Conclusion

In this paper, we have proposed two new sampling-based algorithms to solve the optimal path planning problem in a cost space. They combine the underlying principles of T-RRT and RRT*, the goals being to benefit from their respective strengths and to overcome their respective weaknesses. On the positive side, T-RRT can efficiently explore a cost space thanks to the filtering properties of its transition test, and RRT* is asymptotically optimal. On the negative side, T-RRT is not asymptotically optimal, and RRT* may converge slowly on complex cost spaces. The two hybrid methods are: 1) the Transition-based RRT* (T-RRT*), which is an extension of RRT* integrating the transition test of T-RRT, and 2) the Anytime T-RRT (AT-RRT), which is an extension of T-RRT integrating a useful-cycle addition procedure. We have proven that T-RRT* and AT-RRT are both probabilistically complete and asymptotically optimal. We have evaluated them on several optimal path-planning problems featuring complex, continuous cost functions, and compared them to RRT*. Results show that they converge faster than RRT* toward the optimal path, sometimes even orders of magnitude faster. This is especially true when the search space is very large, when its topology is complex, and/or when dimensionality is high.

Our experiments show that AT-RRT tends to perform slightly better than T-RRT*. As future work, it would be interesting to analyze further how the two algorithms behave, to pinpoint which strategy works best in general, or on particular classes of optimal path-planning problems. Disregarding computational performance, a clear advantage of AT-RRT over T-RRT* is that it can easily be extended into a multiple-tree algorithm, by combining it to the *Multi-T-RRT* [24]. Another interesting aspect of AT-RRT is that it builds a graph containing cycles, therefore providing alternative paths over the space. This could be leveraged when path replanning is required due to errors in the model or moving obstacles.

References

[1] D. Nieuwenhuisen and M. Overmars, "Useful cycles in probabilistic roadmap graphs," in *Proc. IEEE ICRA*, 2004, pp. 446–452.
[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
[3] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *Proc. IEEE ICRA*, 2011, pp. 1478–1483.
[4] J. Marble and K. Bekris, "Asymptotically near-optimal planning with probabilistic roadmap spanners," *IEEE Trans. Robot.*, vol. 29, no. 2, pp. 432–444, Apr. 2013.

[5] A. Dobson and K. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 18–47, 2014.
[6] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proc. IEEE/RSJ IROS*, 2006, pp. 5369–5375.
[7] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Trans. Robot.*, vol. 26, no. 4, pp. 635–646, Aug. 2010.
[8] D. Devaurs, T. Siméon, and J. Cortés, "Enhancing the transition-based RRT to deal with complex cost spaces," in *Proc. IEEE ICRA*, 2013, pp. 4105–4110.
[9] D. Berenson, T. Siméon, and S. Srinivasa, "Addressing cost-space chasms in manipulation planning," in *Proc. IEEE ICRA*, 2011, pp. 4561–4568.
[10] M. Manubens, D. Devaurs, L. Ros, and J. Cortés, "Motion planning for 6-D manipulation with aerial towed-cable systems," in *Proc. RSS*, Berlin, Germany, 2013.
[11] L. Jaillet, F. Corcho, J.-J. Pérez, and J. Cortés, "Randomized tree construction algorithm to explore energy landscapes," *J. Comput. Chem.*, vol. 32, no. 16, pp. 3464–3474, 2011.
[12] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proc.IEEE ICRA*, 1994, pp. 3310–3317.
[13] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer, 1991.
[14] S. LaValle and J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*. Natick, MA, USA: A. K. Peters, 2001, pp. 293–308.
[15] R. Geraerts and M. Overmars, "Creating high-quality paths for motion planning," *Int. J. Robot. Res.*, vol. 26, no. 8, pp. 845–863, 2007.
[16] J. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Proc. IEEE CDC*, 2011, pp. 3276–3282.
[17] R. Geraerts and M. Overmars, "Creating high-quality roadmaps for motion planning in virtual environments," in *Proc. IEEE/RSJ IROS*, 2006, pp. 4355–4361.
[18] Q. Zhu, Y. Wu, G. Wu, and X. Wang, "An improved anytime RRTs algorithm," in *Proc. AICI*, 2009, pp. 268–272.
[19] Y. Abbasi-Yadkori, J. Modayil, and C. Szepesvári, "Extending rapidly-exploring random trees for asymptotically optimal anytime motion planning," in *Proc. IEEE/RSJ IROS*, 2010, pp. 127–132.
[20] R. Alterovitz, S. Patil, and A. Derbakova, "Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning," in *Proc. IEEE ICRA*, 2011, pp. 3706–3712.
[21] R. Luna, I. Şucan, M. Moll, and L. Kavraki, "Anytime solution optimization for sampling-based motion planning," in *Proc. IEEE ICRA*, 2013, pp. 5068–5074.
[22] D. Devaurs, T. Siméon, and J. Cortés, "Efficient sampling-based approaches to optimal path planning in complex cost spaces," in *Proc. WAFR*, Istanbul, Turkey, 2014.
[23] T. Siméon, J.-P. Laumond, and F. Lamiraux, "Move3D: A generic platform for path planning," in *Proc. IEEE ISATP*, 2001, pp. 25–30.
[24] D. Devaurs, T. Siméon, and J. Cortés, "A multi-tree extension of the Transition-based RRT: Application to ordering-and-pathfinding problems in continuous cost spaces," in *Proc. IEEE/RSJ IROS*, 2014, pp. 2991–2996.

Didier Devaurs, photograph and biography not available at the time of publication.

Thierry Siméon, photograph and biography not available at the time of publication.

Juan Cortés, photograph and biography not available at the time of publication.