



HAL
open science

Extensions of sampling-based approaches to path planning in complex cost spaces: applications to robotics and structural biology

Didier Devaurs

► **To cite this version:**

Didier Devaurs. Extensions of sampling-based approaches to path planning in complex cost spaces: applications to robotics and structural biology. Robotics [cs.RO]. INP DE TOULOUSE, 2014. English. NNT: . tel-04261785v1

HAL Id: tel-04261785

<https://theses.hal.science/tel-04261785v1>

Submitted on 4 Nov 2014 (v1), last revised 27 Oct 2023 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Présentée et soutenue par :

Didier Devaurs

le 10/10/2014

Titre :

Extensions of Sampling-based Approaches to
Path Planning in Complex Cost Spaces:
Applications to Robotics and Structural Biology

École doctorale et discipline ou spécialité :

ED MITT : Domaine STIC : Intelligence Artificielle

Unité de recherche :

LAAS-CNRS

Directeur(s) de Thèse :

Juan Cortés, LAAS-CNRS

Jury :

Lydia Kavraki, Rice University, rapporteuse
Frédéric Cazals, INRIA Sophia-Antipolis, rapporteur
Stéphane Redon, INRIA Grenoble, examinateur
Thierry Siméon, LAAS-CNRS, examinateur
Rachid Alami, LAAS-CNRS, examinateur

Abstract

Planning a path for a robot in a complex environment is a crucial issue in robotics. So-called probabilistic algorithms for path planning are very successful at solving difficult problems and are applied in various domains, such as aerospace, computer animation, and structural biology. However, these methods have traditionally focused on finding paths avoiding collisions, without considering the quality of these paths. In recent years, new approaches have been developed to generate high-quality paths: in robotics, this can mean finding paths maximizing safety or control; in biology, this means finding motions minimizing the energy variation of a molecule. In this thesis, we propose several extensions of these methods to improve their performance and allow them to solve ever more difficult problems. The applications we present stem from robotics (industrial inspection and aerial manipulation) and structural biology (simulation of molecular motions and exploration of energy landscapes).

Keywords

path planning · cost space · robotics · structural biology

Acknowledgments

The work presented in this thesis has been supported by:

- the European Community under Contract ICT 287617 “ARCAS”,
- the French National Agency for Research (ANR) under project GlucoDesign,
- the French National Agency for Research (ANR) under project ProtiCAD (project number ANR-12-MONU-0015-01), and
- the HPC-EUROPA project (RII3-CT-2003-506079), with the support of the European Community - Research Infrastructure Action under the FP6 “Structuring the European Research Area” Program.

Summary

Planning a path for a mobile system in a complex environment is a crucial issue in robotics. In this context, sampling-based path planning algorithms have been very successful despite their conceptual simplicity. This is due to their ability to efficiently solve complex planning problems involving mobile systems with numerous degrees of freedom, so-called high-dimensional problems. These algorithms have been successfully used in domains as diverse as robotics, aerospace, manufacturing, virtual prototyping, computer animation, computational structural biology, and medicine. Their underlying principle is to explore the space of the configurations of the mobile system (known as the configuration space) 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. This is referred to as “feasible path planning”. However, in many applications it is important to compute high-quality (i.e. low-cost) paths with respect to a given cost criterion. In recent years, variants of classical sampling-based planners have been developed to take cost criteria into account during the space exploration. This can be referred to as “cost-space path planning”. Some methods even aim at finding the optimal path (i.e. the lowest-cost path) in a cost space, which is referred to as “optimal path planning”.

In this thesis, we propose several extensions of sampling-based path planning algorithms to efficiently solve ever more complex problems. Indeed, many application fields yield increasingly difficult, high-dimensional problems that existing methods cannot cope with, or only with difficulty. Our work focuses on robotics and structural biology applications. On the robotics side, we need planning algorithms for robots moving in large-scale workspaces, such as industrial installations. Beyond the classical computation of a path going from a point A to a point B, industrial inspection tasks require to compute a path going through several waypoints in an efficient manner. Another difficult problem we tackle is that of precise 6-dimensional manipulation performed by a towed-cable system involving three aerial robots. On the structural biology side, all problems are inherently difficult because of their high-dimensionality, even when only small molecules are studied. Our work touches on two different issues: exploring the energy landscape of a small peptide, and simulating the unbinding process of a protein-ligand complex.

The methods presented in this thesis are based on the Rapidly-exploring Random Tree (RRT), a popular algorithm that can solve the feasible path planning problem, as well as some of its variants: the Transition-based RRT (T-RRT), that can solve the cost-space path planning problem, and RRT*, that can solve the optimal path planning problem. This thesis encompasses several algorithmic contributions. First, we present several extensions of T-RRT, allowing for a more efficient cost-space path planning. We improve the original mono-directional variant of T-RRT and propose a bidirectional variant that we generalize into a multiple-tree variant. Second, we combine the concepts underlying T-RRT and RRT* in two different ways, leading to two novel anytime algorithms for optimal path planning with improved results. We also show that this improvement is particularly significant when the topology of the space is complex and/or when its dimensionality is high. Third, we propose three parallel versions of RRT-like algorithms on distributed-memory architectures, which can improve feasible, cost-space and optimal path planning. We evaluate parallel versions of RRT and T-RRT, and we analyze the factors influencing their performance. Finally, we combine several of these approaches (such as the anytime and multiple-tree paradigms) to develop new algorithms able to solve challenging planning problems.

Contents

1	Introduction	9
1.1	Sampling-based Path Planning	9
1.2	Algorithmic Contributions of the Thesis	11
1.3	Applications	13
2	Related Sampling-based Methods for Path Planning	15
2.1	Feasible Path Planning	15
2.1.1	Theoretical Framework	15
2.1.2	Rapidly-exploring Random Tree (RRT)	16
2.1.3	Probabilistic Road-Map (PRM)	17
2.2	Cost-Space Path Planning	18
2.2.1	Threshold-based RRT (RRT_{obst})	19
2.2.2	Transition-based RRT (T-RRT)	20
2.3	Optimal Path Planning	22
2.3.1	Theoretical Framework	22
2.3.2	Path-Quality Criteria	23
2.3.3	The RRT* Algorithm	24
2.4	Parallel Path Planning	26
2.4.1	General Parallel Approaches	26
2.4.2	Parallel RRT	26
3	Efficient Cost-based Path Planning in Complex Continuous Cost Spaces	29
3.1	Extensions of the Mono-directional Variant of T-RRT	29
3.1.1	Path Planning Problems and Evaluation Settings	30
3.1.2	Improvement of the Transition Test	31
3.1.3	Connect and Goal-biased Extensions of T-RRT	33
3.2	Bidirectional Extension of T-RRT	33
3.2.1	The Bidirectional T-RRT Algorithm	35
3.2.2	Evaluation and Analysis	36
3.2.3	Industrial Inspection Problem	39
3.3	Multiple-Tree Extension of T-RRT	40
3.3.1	Scope of the Approach	40
3.3.2	The Multi-T-RRT Algorithm	41
3.3.3	Other Multi-Tree Variants	42
3.3.4	Evaluation Results	42
3.3.5	Useful-Cycle Addition	45
3.4	Conclusion	46

4	Efficient Optimal Path Planning in Complex Continuous Cost Spaces	49
4.1	Algorithms	50
4.1.1	Transition-based RRT* (T-RRT*)	50
4.1.2	Anytime Transition-based RRT (AT-RRT)	52
4.2	Theoretical Analysis	53
4.3	Evaluation	54
4.3.1	Path Planning Problems	54
4.3.2	Settings	56
4.3.3	Results	57
4.4	Conclusion	60
5	Parallel Path Planning on Distributed-Memory Architectures	63
5.1	Parallelization of RRT	64
5.1.1	OR Parallel RRT	64
5.1.2	Collaborative Building of a Single RRT	65
5.1.3	Implementation Framework	67
5.2	Experiments with RRT and T-RRT	68
5.2.1	Path Planning Problems and Evaluation Settings	68
5.2.2	First Experiment - High Expansion Cost	68
5.2.3	Second Experiment - Variable Expansion Cost	69
5.2.4	Robotic Examples	72
5.3	Analysis of the Parallel Algorithms	72
5.3.1	OR Parallel RRT	72
5.3.2	Distributed RRT	74
5.3.3	Manager-Worker RRT	75
5.3.4	Discussion	76
5.4	Application to Other RRT-like Algorithms	77
5.5	Conclusion	77
6	Application to Complex Robotic Problems	79
6.1	6-D Manipulation with an Aerial Towed-Cable System	80
6.1.1	Path Planning Approach	81
6.1.2	Test Cases	83
6.1.3	Conclusion	86
6.2	Ordering-and-Pathfinding Problems	88
6.2.1	The Anytime Multi-T-RRT Algorithm	88
6.2.2	Application to Industrial Inspection	89
6.2.3	Conclusion	91
7	Application to Structural Biology Problems	93
7.1	Exploration of the Energy Landscape of a Small Peptide	94
7.1.1	Methods	94
7.1.2	Results and Discussion	97
7.1.3	Conclusion	107
7.2	Simulation of Protein-Ligand Unbinding	108
7.2.1	Methods	109
7.2.2	Results and Discussion	110
7.2.3	Conclusion	113

8	Conclusion	115
8.1	Summary of the Algorithmic Contribution	115
8.2	Summary of the Applications	116
8.3	Directions of Future Research	116
A	French Summary	119
A.1	Introduction	121
A.1.1	La Planification de Chemin par Échantillonnage	121
A.1.2	Contributions Algorithmiques de la Thèse	123
A.1.3	Applications	125
A.2	Méthodes Apparentées de Planification de Chemin	127
A.2.1	Planification de Chemin Faisable	127
A.2.2	Planification de Chemin en Espace de Coût	127
A.2.3	Planification de Chemin Optimal	128
A.2.4	Planification de Chemin en Parallèle	129
A.3	Planification de Chemin Basée-coût Efficace	129
A.3.1	Extensions de la Variante Mono-directionnelle de T-RRT	130
A.3.2	Extension Bidirectionnelle de T-RRT	130
A.3.3	Extension Multi-arbres de T-RRT	130
A.4	Planification de Chemin Optimal Efficace	131
A.4.1	Algorithmes	132
A.4.2	Analyse Théorique	132
A.4.3	Évaluation	132
A.5	Planification de Chemin en Parallèle	132
A.5.1	Parallélisation de RRT	133
A.5.2	Expérimentations avec RRT et T-RRT	133
A.5.3	Analyse des Algorithmes Parallèles	134
A.5.4	Application à d’Autres Algorithmes de Type RRT	134
A.6	Application à des Problèmes de Robotique Complexes	134
A.6.1	Manipulation 6-D avec un Système à Câbles Aérien	134
A.6.2	Problèmes “d’Ordonnancement et de Recherche de Chemin”	134
A.7	Application à des Problèmes de Biologie Structurale	135
A.7.1	Exploration du Paysage Énergétique d’un Petit Peptide	135
A.7.2	Simulation de la Séparation d’un Complexe Protéine-Ligand	135
A.8	Conclusion	136
A.8.1	Résumé des Contributions Algorithmiques	136
A.8.2	Résumé des Contributions Applicatives	136
A.8.3	Perspectives de Travaux Futurs	137
	Bibliography	139

Chapter 1

Introduction

Planning a path for a mobile system, such as a robot, in a complex environment (typically, the physical world) is a crucial issue in robotics. Addressing this issue has potential repercussions reaching far beyond the field of robotics. We can start by mentioning the application domains that directly profit from technological developments in robotics, such as the aerospace and manufacturing industries. Other application domains that are less obviously connected to robotics can also benefit, as is the case for virtual prototyping using CAD/CAM software, for instance. Indeed, the (dis)assembly tests performed in this field can be modeled as path planning tasks. Another example is computer animation: if virtual actors are modeled as robots, path planning techniques become graphic animation tools. Furthermore, medical applications also exist: for example, finding a minimally-invasive path for a surgical tool, given a 3-dimensional model of a patient’s body, can be seen as a path planning problem. Finally, thanks to the similarities between structural models of robots and molecules, path planning techniques can be applied in computational structural biology.

The path planning problem was originally formulated for a rigid-bodied robot having to move in an environment containing static obstacles, while only having to avoid collisions. This has usually been referred to as the “piano mover’s problem”. A geometrical formulation of the path planning problem was derived from the definition of the *configuration space*, i.e. the space of all the possible configurations of the mobile system. Based on this concept, the first methods proposed to solve the path planning problem were deterministic algorithms providing an exact solution. Nevertheless, these methods cannot cope with difficult problems, such as those mentioned above. The difficulty of a path planning problem stems from the complexity of the mobile system, which is mainly expressed through the number of its degrees of freedom, as well as the complexity of potential additional constraints. Path planning problems involving mobile systems characterized by numerous degrees of freedom are usually referred to as “high-dimensional problems”.

1.1 Sampling-based Path Planning

Contrary to deterministic algorithms, so-called “probabilistic” methods for path planning have been successful at efficiently solving complex, high-dimensional problems. Their underlying principle is to explore the configuration space of the mobile system by sampling it, and to build a graph representing the connectivity of this space. Despite their conceptual simplicity, sampling-based path planners have proven valuable to a wide range of applications, such as those mentioned above. As a result, they have benefited from a considerable research effort during the last 15 years. Several methods have been proposed, and have then been extended to deal with challenging issues, such as kinodynamic planning, loop-closure constraints, or dynamic environments. Among these methods, the Rapidly-exploring Random Tree (RRT) algorithm has become very popular.

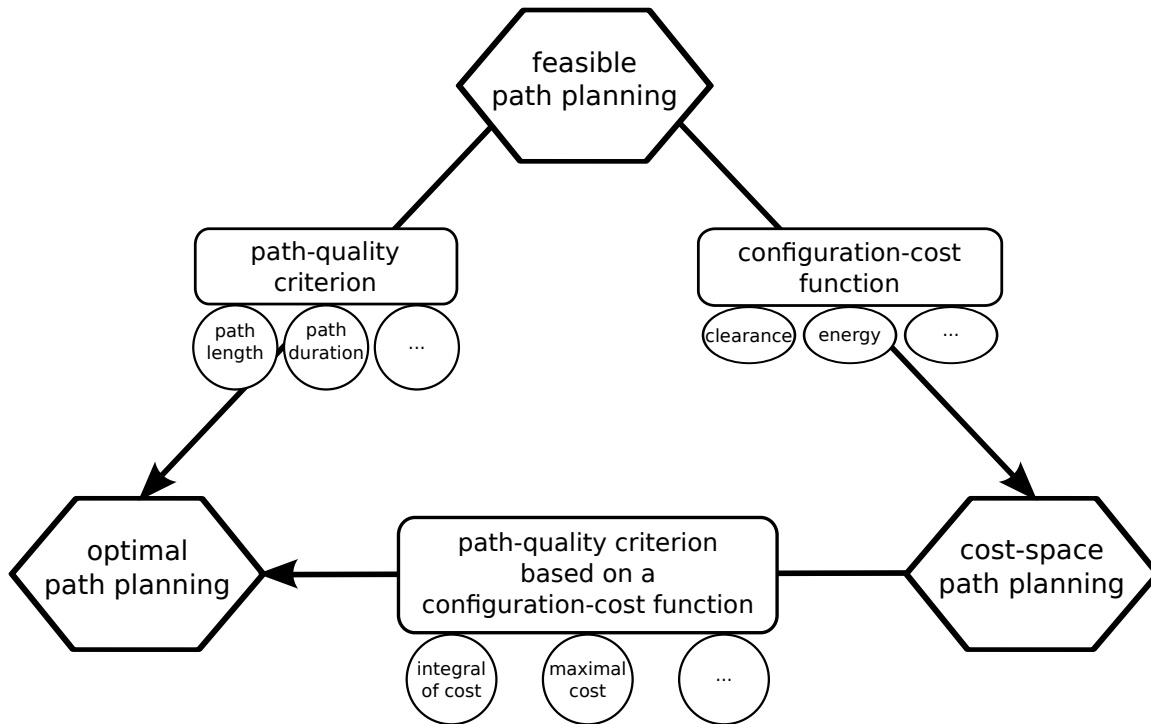


Figure 1.1: Schematic representation of the relationships that exist between the three path planning paradigms addressed in this thesis.

Sampling-based path planners such as RRT have traditionally been used to find feasible paths (i.e. collision-free paths) without considering the quality of these paths. This paradigm can be referred to as “feasible path planning” (cf. Fig. 1.1). However, in many application fields it is important to compute high-quality paths, with respect to a given quality criterion. Historically, the first quality criteria to be considered were path length and path duration, mainly to address the fact that the paths produced by sampling-based algorithms were usually “jerky”. Later on, the notion of clearance was introduced: the underlying idea was to generate paths along which the mobile system would remain as far as possible from the obstacles, to ensure its own safety. When using such quality criteria is desirable, after a solution path has been computed by a sampling-based path planner, it is most common to try and improve the quality of this path during a post-processing phase involving so-called “smoothing” methods. Nevertheless, such methods only allow to improve the path locally. We explain, later on, how better results can be achieved by taking the quality criteria into account during the exploration of the configuration space.

As already mentioned, it can be useful to generate high-quality paths, based on the use of specific quality criteria. In some application contexts, instead of considering a criterion, such as path length, that assesses the quality of a path as a whole, it might be more interesting to ensure that all configurations along the path have low costs, with respect to a given cost function. For instance, when high-clearance paths are desirable, the cost of a configuration can be based on the inverse of the distance between the mobile system and the closest obstacle. In structural biology, the cost of a conformation of a molecule is the molecular energy. When such a cost function is defined on the configuration space, we call the latter a “cost space”. This paradigm can thus be referred to as “cost-space path planning” (cf. Fig. 1.1). In recent years, variants of classical sampling-based algorithms have been developed to take cost functions into account during the exploration of the configuration space. One of these methods is a variant of RRT called the Transition-based RRT (T-RRT).

Beyond the computation of a high-quality path, one might be looking for an optimal path, with respect to a given path-quality criterion. This paradigm is referred to as “optimal path planning” (cf. Fig. 1.1). When applied to this problem, classical grid-based methods, such as A* or D*, can compute resolution-optimal solution paths. 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 criterion. For instance, the *visibility diagram* produces the shortest path, and the *Voronoi diagram* generates the path with optimal clearance. Nevertheless, such methods are also limited to low-dimensional spaces and can only deal with polygonal obstacles. On the other hand, classical sampling-based path planners can cope with high-dimensional spaces, but they usually produce sub-optimal solutions. As a complement, the aforementioned smoothing methods can be used to improve path quality in a post-processing phase, but they offer no guarantee of converging toward the global optimum. The first RRT-like path planner providing such guarantee was RRT*.

Even though classical sampling-based path planners have achieved great success, their cost-space counterparts are still in their infancy and suffer from strong limitations. Indeed, many application fields yield increasingly difficult, high-dimensional problems that existing methods cannot really cope with, or only with difficulty. Besides, optimal path planning is even more challenging on such problems, as it amounts to solving a non-linear, non-convex, high-dimensional, global optimization problem. Additionally, the most commonly used path-quality criteria are still path length and path duration, and the most commonly used configuration-cost functions are discrete, coarse-grained ones. Our work aims at dealing with continuous configuration-cost functions, as well as path-quality criteria based on these functions (cf. Fig. 1.1), which is more challenging. Our goals are to improve the capabilities and the performance of sampling-based algorithms, mainly in the context of cost-space and optimal path planning. We also aim to study applications that require the creation of new path planning methods, in the fields of robotics and structural biology.

1.2 Algorithmic Contributions of the Thesis

In this thesis, we propose several extensions of sampling-based path planning algorithms to efficiently solve ever more difficult problems involving complex continuous cost functions. The methods we develop are based on RRT, which can solve the feasible path planning problem, as well as some of its variants: T-RRT, that can solve the cost-space path planning problem, and RRT*, that can solve the optimal path planning problem. The RRT, T-RRT and RRT* algorithms are reviewed in Chapter 2. Then, we present the various algorithmic contributions proposed in this thesis (cf. Fig. 1.2).

First, with the objective of achieving a more efficient cost-space path planning, we develop several extensions of T-RRT, starting from enhancements of its original mono-directional variant, then presenting a bidirectional variant, and finally generalizing it into a multiple-tree variant (cf. Fig. 1.2 and Chapter 3). More precisely, we suggest to improve the performance of the mono-directional T-RRT (that was originally proposed as an extension of the basic *Extend* RRT) by modifying the implementation of its transition test (cf. Section 3.1). We also show that using the *Connect* T-RRT or the *Goal-biased* T-RRT can generally provide improvements (cf. Section 3.1). Then, we present a bidirectional extension of T-RRT that reduces running time and sometimes increases (or otherwise maintains) solution-path quality (cf. Section 3.2). By generalizing this approach, we also develop a multiple-tree extension of T-RRT that can compute a path going through a set of waypoints, and we show that it outperforms path planners involving the *Bidirectional* T-RRT (cf. Section 3.3). Throughout Chapter 3, we apply these extensions of T-RRT to simulated yet practical robotic problems, such as industrial inspection tasks involving aerial robots.

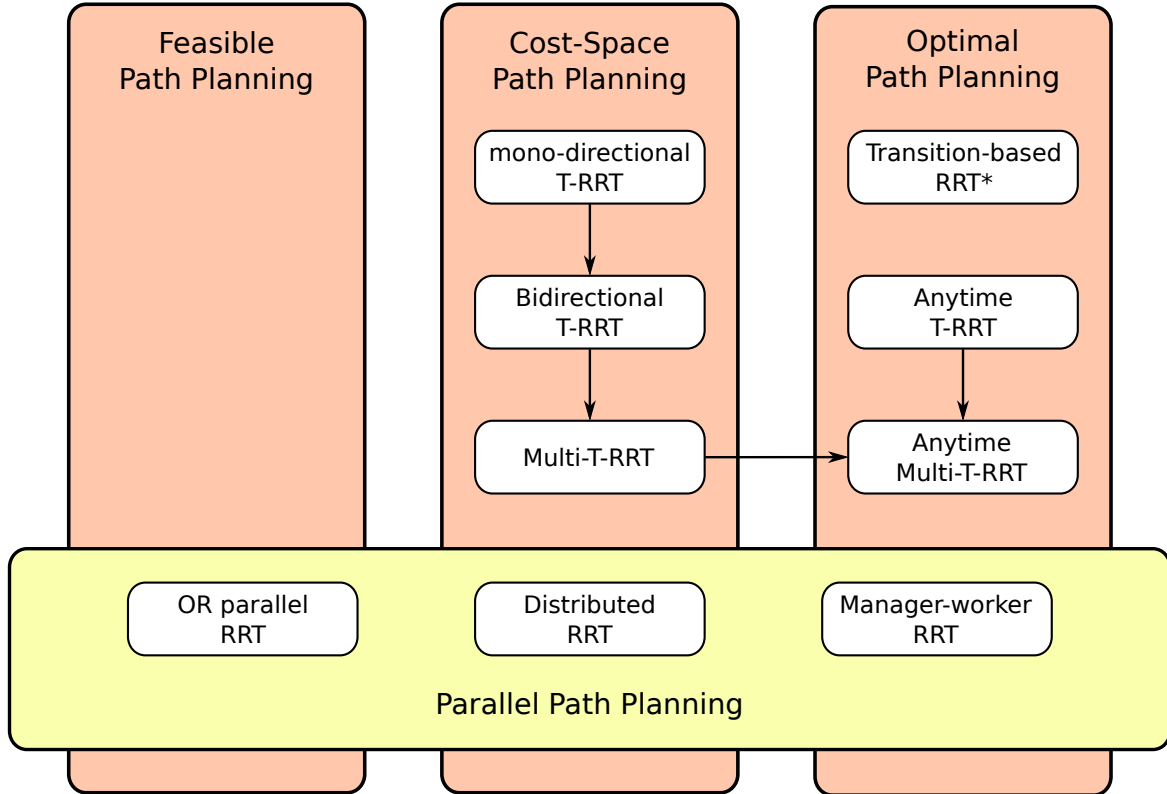


Figure 1.2: Schematic organization of the principal algorithms proposed in this thesis, and representation of the relationships between them.

Second, to improve efficiency in the context of optimal path planning, we combine the beneficial, underlying concepts of T-RRT and RRT* in two different ways (cf. Chapter 4). We propose an extension to RRT* named *Transition-based RRT** (T-RRT*) and an extension to T-RRT named *Anytime T-RRT* (AT-RRT). From their definitions, T-RRT* and AT-RRT feature both 1) the cost-based filtering properties of the transition test of T-RRT, favoring the creation of new nodes in low-cost regions of the configuration space, and 2) the quality-based management of edges of RRT*, allowing the quality of the solution path to increase with time (cf. Section 4.1). We show that both algorithms are probabilistically complete and asymptotically optimal (cf. Section 4.2). Then, we evaluate T-RRT* and AT-RRT on several optimal path-planning problems, and show that they converge toward the optimal solution-path faster than RRT* (cf. Section 4.3). We also show that the performance improvement they achieve is particularly significant when the topology of the configuration space is complex and/or when its dimensionality is high.

Third, to improve the efficiency of feasible, cost-space and optimal path planning, we propose three parallelization strategies for RRT-like algorithms (cf. Fig. 1.2 and Chapter 5). We focus on parallelizing RRT on large-scale distributed-memory architectures, which requires using the Message Passing Interface (MPI). More precisely, we develop three parallel versions of RRT, based on classical parallelization schemes: *OR parallel RRT*, *Distributed RRT* and *Manager-worker RRT* (cf. Section 5.1). Then, we evaluate the parallel versions of RRT and T-RRT on several path planning problems (cf. Section 5.2), and we analyze the various factors influencing their performance (cf. Section 5.3). Our evaluation results show that parallelizing RRT-like algorithms with MPI can provide substantial performance improvement in several cases that occur in numerous complex robotic problems and all structural biology problems. Additionally, we discuss how the RRT-like algorithms introduced in this thesis can be parallelized (cf. Section 5.4).

Finally, we combine several of the approaches introduced in this thesis to develop novel algorithms that can solve new kinds of challenging path-planning problems. Our motivation for presenting these methods separately in Chapters 3 to 5 was the will to evaluate and analyze them independently of each other. On the other hand, interesting new applications can be addressed when combining some of these methods together. For instance, we integrate the anytime and multiple-tree paradigms to create an *Anytime Multi-T-RRT* (cf. Fig. 1.2) that we use in robotics (cf. Chapter 6) and structural biology (cf. Chapter 7).

1.3 Applications

In addition to the aforementioned algorithmic contributions, this thesis also features several contributions on the application side. Indeed, we have applied several of the sampling-based path-planning techniques presented in this thesis to challenging (and sometimes new) path-planning problems encountered in robotics (cf. Chapter 6) and computational structural biology (cf. Chapter 7).

Robotics

On the robotics side, we need path-planning methods that can deal with robots moving in large-scale workspaces, such as industrial installations (oil platform, power plant, steel factory, etc). Beyond the classical computation of a path going from a point A to a point B (which we refer to as the “init-to-goal” problem), industrial inspection tasks require to compute a path going through a given set of waypoints in an efficient manner, which we refer to as the “ordering-and-pathfinding problem” (cf. Section 6.2). To solve this hybrid task-and-path planning problem, we propose a variant of T-RRT called *Anytime Multi-T-RRT*, based on the combination of two extensions of T-RRT presented in this thesis: the *Multi-T-RRT* and the *Anytime T-RRT*. Using the *Anytime Multi-T-RRT*, ordering-and-pathfinding problems can be solved from a purely geometrical perspective, without having to use a symbolic task planner. We demonstrate this method on a simulated industrial inspection problem involving an aerial robot.

Another difficult problem we tackle here is that of precise 6-dimensional manipulation performed by a towed-cable system involving cooperative aerial robots (cf. Section 6.1). For that, we propose a system, that we have called *FlyCrane*, and that consists of a platform attached to three flying robots using six fixed-length cables. In addition, the main element of our approach is an application-specific configuration-cost function that takes the constraints inherent to this robotic system into account. More precisely, this cost function is based on wrench-feasibility constraints (derived from the static analysis of cable-driven manipulators) and on thrust constraints. To validate our approach, we study two simulated manipulation problems. Our experiments show the superiority of cost-space path planning over feasible path planning when dealing with such complex robotic systems.

Most of the path-planning problems we address on the robotics side emanate from the ARCAS European project¹. Some of the goals of this project are to develop robotic systems involving cooperative aerial robots for the installation, the inspection, and the maintenance of industrial installations in places that are difficult to access for humans. An example of possible application is the construction of landing platforms in uneven terrains, for manned or unmanned aircrafts. Another example is the assembly of temporary structures for the evacuation of people in rescue operations. Note that, despite the interest of tackling such problems in real-life situations, the role of our team has been to develop and evaluate new path-planning methods within simulated environments. Real-life experiments are part of the ongoing work within the ARCAS project.

¹<http://www.arcas-project.eu>

Computational Structural Biology

On the computational structural biology side, all problems are inherently difficult because of their high dimensionality, even when only small molecules are considered.

The first issue we address in this thesis is the exploration of the energy landscapes of small yet highly-flexible peptides (cf. Section 7.1). For that, we combine two complementary sampling-based methods. 1) We propose a simplified version of the *Basin Hopping* algorithm that can quickly reveal the meta-stable structural states of a peptide. In computational structural biology, Basin Hopping is a classical method that enables sampling local minima on the energy landscape of a molecule. 2) Then, we use the *Multi-T-RRT* and the *Anytime Multi-T-RRT* to quickly determine transition state and transition path ensembles, as well as transition probabilities between these meta-stable states. We validate this combined approach on the terminally-blocked alanine.

The second issue we address is the simulation of the unbinding process of a protein-ligand complex (cf. Section 7.2). We propose an approach that builds on a mechanistic representation of the molecular system and that currently considers only partial flexibility. Besides, at the moment, the approach is purely geometric, which means that no molecular energy is computed and that motions are validated only on the basis of collision avoidance. This simplification of the problem allows us to use a variant of RRT called *Manhattan-like* RRT (ML-RRT), whose exploratory efficiency leads to very short computing times. This was a requirement imposed by our decision to implement this method as an efficient web application. This tool yields ligand unbinding pathways that, as a first approximation, can provide useful information about protein-ligand interactions. We demonstrate this approach on the hexameric insulin-phenol complex. Finally, let us mention that integrating molecular energy computations into this approach is part of our ongoing research.

Our work in computational structural biology has been carried out within two research projects named GlucoDesign and ProtiCAD. The objective of the GlucoDesign project was the computer-aided design of enzymatic glycosylation tools for the synthesis of vaccines against endemic shigellosis (or bacillary dysentery). The goals of the ProtiCAD² project are to yield advances in a general methodology for protein design, and to develop computational tools for the synthesis of new proteins.

²<http://projects.laas.fr/ProtiCAD>

Chapter 2

Related Sampling-based Methods for Path Planning

Sampling-based algorithms for path planning have been very successful at efficiently solving difficult planning problems involving mobile systems characterized by numerous degrees of freedom, so-called high-dimensional problems [33, 104]. They have proven valuable in a wide range of application domains, such as robotics, aerospace, manufacturing, virtual prototyping, computer animation, medicine, and computational structural biology. As a result, they have benefited from a considerable research effort during the last 15 years. Several approaches have been proposed (see, for example, [75, 95, 106]) and then extended to deal with challenging issues, such as kinodynamic planning [40, 75, 105], loop-closure constraints [38, 160], or dynamic environments [59, 61, 67, 152, 166].

We now quickly review some of these existing methods, within the contexts of feasible, cost-space, and optimal path planning. We focus more specifically on the algorithms that we extend in this thesis or that we use for comparison with our approaches in experimental evaluations. We also review related work in the context of parallel path planning.

2.1 Feasible Path Planning

Traditionally, path planning has focused on computing feasible paths for a mobile system in an environment containing obstacles [102]. Informally speaking, for a path to be feasible, it has to avoid collisions with obstacles and collisions between articulated parts of the mobile system (so-called self-collisions).

2.1.1 Theoretical Framework

The classical formulation of the path planning problem relies on abstracting the workspace of a mobile system into a *configuration space* \mathcal{C} , also called \mathcal{C} -space [110]. A configuration $q \in \mathcal{C}$ describes the position and volume occupied by the mobile system in the workspace. The subset of \mathcal{C} containing the configurations inducing self-collisions or collisions with some obstacles in the workspace is denoted by $\mathcal{C}_{\text{obst}}$. 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 $cl(\cdot)$ 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 the \mathcal{C} -space, and let Π_{free} denote 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 [93]. This property is based on the concept of *strong clearance*. Given $\lambda \in \mathbb{R}_+$, a path $\pi \in \Pi_{\text{free}}$ is said to have strong λ -clearance if π lies entirely inside the λ -interior of $\mathcal{C}_{\text{free}}$. The λ -interior of $\mathcal{C}_{\text{free}}$ is the set of configurations that are at least a distance λ away from any configuration in $\mathcal{C}_{\text{obst}}$. From that, robust feasibility is defined as follows:

Definition 2 (Robust feasibility). *A path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$ is said to be robustly feasible if there exists a path $\pi \in \Pi_{\text{feas}}$ having strong λ -clearance, for some $\lambda \in \mathbb{R}_+$.*

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 providing an exact solution [102]. These methods 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.

In this thesis, we focus on sampling-based approaches because they can solve the complex, high-dimensional problems we are dealing with. The underlying principle of these methods is to explore the configuration space of the mobile system by sampling it, and to build a graph representing the connectivity of this space. The most popular sampling-based path planners, that we present in the next sections, are the Rapidly-exploring Random Tree (RRT) and the Probabilistic Road-Map (PRM). As all sampling-based path planners, they are not complete, but they satisfy a property called *probabilistic completeness*, that can be interpreted as a notion of “almost-sure” success [33]:

Definition 3 (Probabilistic completeness). *An algorithm \mathcal{A} is said to be 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.*

2.1.2 Rapidly-exploring Random Tree (RRT)

The Rapidly-exploring Random Tree (RRT) is a prevalent sampling-based algorithm, usually applied to single-query path planning problems [100, 103, 106]. It is suited to solve robot path planning problems involving holonomic, nonholonomic, kinodynamic, or kinematic loop-closure constraints [38, 105, 106]. It is also applied to planning in discrete spaces or for hybrid systems [23, 56]. In computational biology, it is used to analyze genetic network dynamics [9] or protein-ligand interactions [39], for instance.

Let us consider 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} that tends to rapidly expand over the \mathcal{C} -space, thanks to the implicit enforcement of a Voronoi bias [106]. The nodes and edges of \mathcal{T} correspond to configurations and local moves between configurations, respectively. The pseudo-code of the original variant of RRT is presented in Algorithm 1. At each iteration, a configuration q_{rand} is randomly sampled in \mathcal{C} . Then, an expansion toward q_{rand} is attempted, starting from its nearest neighbor, q_{near} , in \mathcal{T} . If the expansion 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 the exploration 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.

Algorithm 1: RRT

```

input : the feasible path planning problem  $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$ 
output: the tree  $\mathcal{T}$ 
1  $\mathcal{T} \leftarrow \text{initTree}(q_{\text{init}})$ 
2 while not stoppingCriteria( $\mathcal{T}$ ) do
3    $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
4    $q_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}, q_{\text{rand}})$ 
5    $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}, q_{\text{rand}})$ 
6   if  $q_{\text{new}} \neq \text{null}$  then
7      $\text{addNewNode}(\mathcal{T}, q_{\text{new}})$ 
8      $\text{addNewEdge}(\mathcal{T}, q_{\text{near}}, q_{\text{new}})$ 
9 return  $\mathcal{T}$ 

```

The expansion procedure involved in the main loop of the RRT algorithm (line 5) can be implemented in two different manners, called *Extend* and *Connect*. The *Extend* procedure consists of performing an interpolation between q_{near} and q_{rand} , at a distance equal to the extension step-size, δ , from q_{near} (except if $\text{distance}(q_{\text{near}}, q_{\text{rand}}) < \delta$, in which case the result of the interpolation is q_{rand} itself) [103]. The *Connect* procedure consists of iterating the Extend function until q_{rand} is reached, or until an obstacle is encountered, in which case q_{new} is defined as the last collision-free configuration [100]. Note that, when using the Connect method, it is also possible to add all intermediate configurations (produced by the consecutive Extend procedures) as nodes in \mathcal{T} .

Theorem 1 (Probabilistic completeness of RRT). *The RRT algorithm is probabilistically complete [106].*

Despite its successes, when applied to complex problems, the growth of an RRT can become computationally expensive [29, 36, 84, 162]. Some techniques have been proposed to improve the efficiency of RRT, by dynamically controlling its sampling domains [84], reducing the dispersion of the samples drawn in \mathcal{C} [109], or reducing the complexity of the nearest neighbor search [162]. Performance can also be improved by relaxing the constraints used to validate local motions and using gap reduction techniques perturbing the solution path in a post-processing phase [29]. A well-known weakness of RRT is its sensitivity to the metric defined on the \mathcal{C} -space; this issue has been addressed in [30]. Furthermore, a resolution complete variant of RRT has been developed [31]. In this thesis, we present an additional enhancement of RRT, which consists of parallelizing it on large-scale distributed-memory architectures (cf. Chapter 5).

2.1.3 Probabilistic Road-Map (PRM)

The Probabilistic Road-Map (PRM) is another well-known sampling-based algorithm, usually applied to multiple-query path planning problems [95]. There exist numerous variants of PRM. The pseudo-code of the one used in this thesis is presented in Algorithm 2. Given the feasible path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$, starting from the initial configuration q_{init} , this version of PRM iteratively builds a graph \mathcal{G} over the \mathcal{C} -space. The most standard versions of PRM differ from this variant in that they consist of two distinct phases: 1) a set of configurations is randomly sampled in $\mathcal{C}_{\text{free}}$, and 2) connections are created between these configurations. However, interleaving these two phases and building the graph iteratively, as is done in Algorithm 2, allows us to benefit from an anytime behavior that is required by the experimental evaluation in which PRM is involved here (cf. Section 3.3).

Algorithm 2: PRM

```

input : the feasible path planning problem  $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$ 
         the connection radius  $R$ 
output: the graph  $\mathcal{G}$ 
1  $\mathcal{G} \leftarrow \text{initGraph}(q_{\text{init}})$ 
2 while not stoppingCriteria( $\mathcal{G}$ ) do
3    $q_{\text{new}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C}_{\text{free}})$ 
4   addNewNode( $\mathcal{G}, q_{\text{new}}$ )
5    $Q_{\text{near}} \leftarrow \text{findNodesInBall}(\mathcal{G}, q_{\text{new}}, R)$ 
6   foreach  $q_{\text{near}} \in Q_{\text{near}}$  do
7     if isCollisionFree(path( $q_{\text{new}}, q_{\text{near}}$ )) then
8       addNewEdge( $\mathcal{G}, q_{\text{new}}, q_{\text{near}}$ )
9 return  $\mathcal{G}$ 

```

The variant of PRM we use works as follows (see Algorithm 2): At each iteration, a new configuration q_{new} is randomly sampled in $\mathcal{C}_{\text{free}}$, and a new node is added to \mathcal{G} . Then, all the nodes in \mathcal{G} that are within a distance R of q_{new} (where R is called the connection radius) are considered for edge creation. For each candidate q_{near} , if the path between q_{new} and q_{near} is collision-free, a new edge is added to \mathcal{G} . The criteria on when to stop the exploration are the same as those defined for RRT.

Theorem 2 (Probabilistic completeness of PRM). *The PRM algorithm is probabilistically complete [95].*

2.2 Cost-Space Path Planning

Instead of building paths that are only feasible, it can be important to generate “high-quality” paths, with respect to some quality criteria. Historically, because the paths produced by sampling-based algorithms were usually “jerky”, the first quality criteria to be used were path length and path duration, which assess the quality of a path as a whole [69]. However, it may be more interesting to ensure that all configurations along the path have low costs, with respect to a given cost function. When such a cost function is defined on the \mathcal{C} -space, we call the latter a “cost space”, and we talk about “cost-space path planning”.

Early work in cost-space path planning only involved discrete, coarse-grained cost functions [60, 93]. Our work focuses on continuous 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 a robot within a 2-D terrain. In planning problems where high-clearance paths are desirable, the cost of a configuration can be based on the inverse of the distance between the mobile system and the closest obstacle [49, 83]. More complex cost functions can appear in robotic problems [12, 116] and structural biology problems [82].

Let $c : \mathcal{C} \rightarrow \mathbb{R}_+$ denote a 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 performing a rejection sampling of configurations in \mathcal{C} , by filtering configurations on the basis of 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.

Algorithm 3: RRT_{obst}

```

input : the cost-space path planning problem  $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c)$ 
         the cost threshold  $c_{\text{max}}$ 
         the increase rate of the cost threshold  $c_{\text{rate}}$ 
output: the tree  $\mathcal{T}$ 
1  $\mathcal{T} \leftarrow \text{initTree}(q_{\text{init}})$ 
2  $c_{\text{max}} \leftarrow c(q_{\text{init}})$ 
3 while not stoppingCriteria( $\mathcal{T}$ ) do
4    $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
5    $q_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}, q_{\text{rand}})$ 
6    $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}, q_{\text{rand}})$ 
7   if  $q_{\text{new}} \neq \text{null}$  and  $c(q_{\text{new}}) < c_{\text{max}}$  then
8      $\text{addNewNode}(\mathcal{T}, q_{\text{new}})$ 
9      $\text{addNewEdge}(\mathcal{T}, q_{\text{near}}, q_{\text{new}})$ 
10   $c_{\text{max}} \leftarrow c_{\text{max}} + c_{\text{rate}}$ 
11 return  $\mathcal{T}$ 

```

The first approaches dealing with cost-space path planning were based on RRT. Unfortunately, they were all focused on specific applications in the area of 2D robot navigation [57, 58, 60, 61, 107, 151], and some of them were evaluated only on configuration spaces involving very coarse-grained, discrete cost functions [60, 61, 151]. More importantly, all these methods suffer from different practical drawbacks [83]. For example, some of them rely on the estimated *cost-to-goal*, which tends to bias the search straight toward the goal at the expense of higher-quality paths [60, 61, 151]. Also, the threshold-based method presented in [57, 58] suffers from the non-decreasing nature of its threshold and from its high sensitivity to the increase rate of the threshold [83]. As it is involved in some of our experiments, we now present in greater details this threshold-based method. After that, we present a variant of RRT called the Transition-based RRT (T-RRT) that has been more successful for cost-space path planning than the aforementioned methods.

2.2.1 Threshold-based RRT (RRT_{obst})

The RRT_{obst} algorithm is an extension of RRT devised specifically for cost-space path planning. It was proposed in the context of rough terrain navigation [58]. The cost function introduced in that work aimed to assess the level of difficulty (the so-called “obstacleness”) corresponding to attaining a given configuration of the robot. The pseudo-code of RRT_{obst} is presented in Algorithm 3. The idea is to accept or reject new configurations created by the RRT expansion based on their costs: a configuration is accepted if its cost is below a given threshold, denoted by c_{max} , and rejected otherwise. This cost threshold is a dynamic parameter of the algorithm: it is initialized to a low value (typically the cost of the initial configuration) and iteratively increased during the space search, using the cost increment c_{rate} . As a result, the search performed by RRT_{obst} is biased toward low-cost regions of the configuration space. Higher-cost regions of the space are explored only when inevitable. Therefore, paths produced by RRT_{obst} follow low-cost regions of the space, and their maximal cost is kept relatively low.

In our experimental evaluations (cf. Section 3.3), we use a multiple-tree variant of RRT_{obst} , called $\text{RRT}_{\text{obst way}}$ [57]. This method grows several trees rooted at various waypoints on the \mathcal{C} -space, and regularly tries to connect them. The idea is to explore significant low-cost basins of the cost landscape before connections are made between them.

Algorithm 4: Transition-based RRT

```

input : the cost-space path planning problem  $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c)$ 
output: the tree  $\mathcal{T}$ 
1  $\mathcal{T} \leftarrow \text{initTree}(q_{\text{init}})$ 
2 while not stoppingCriteria( $\mathcal{T}$ ) do
3    $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
4    $q_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}, q_{\text{rand}})$ 
5   if refinementControl( $\mathcal{T}, q_{\text{near}}, q_{\text{rand}}$ ) then
6      $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}, q_{\text{rand}})$ 
7     if  $q_{\text{new}} \neq \text{null}$  and transitionTest( $\mathcal{T}, c(q_{\text{near}}), c(q_{\text{new}})$ ) then
8       addNewNode( $\mathcal{T}, q_{\text{new}}$ )
9       addNewEdge( $\mathcal{T}, q_{\text{near}}, q_{\text{new}}$ )
10 return  $\mathcal{T}$ 

```

2.2.2 Transition-based RRT (T-RRT)

The Transition-based RRT (T-RRT) algorithm is a general sampling-based approach to cost-space path planning that can deal with any configuration-cost function [83]. Being an extension of RRT, T-RRT combines the exploratory strength of RRT with a stochastic optimization mechanism. It has been successfully applied to various cost-space path-planning problems in robotics [12, 79, 83, 114] (some examples even involving human–robot interactions [114]) and in computational structural biology [79, 82]. When compared to methods developed earlier [58, 151], T-RRT produced better-quality paths [83].

T-RRT extends RRT by integrating a stochastic transition test enabling it to favor the exploration of low-cost regions of the \mathcal{C} -space [83]. This transition test is based on the Metropolis criterion typically used in Monte Carlo optimization methods [65]. These techniques aim at finding global minima in complex spaces and involve randomness as a way to avoid being trapped in local minima. Similarly, T-RRT uses a transition test to accept or reject a new candidate configuration, based on the cost variation associated with the local motion whose target is the new configuration and whose source is its nearest neighbor in the tree. The pseudo-code of T-RRT (shown in Algorithm 4) is analogous to that of the *Extend* RRT, with the addition of the `transitionTest` and `refinementControl` functions.

The `transitionTest` presented in Algorithm 5 is used to accept or reject the move between two configurations on the basis of their costs c_i and c_j . Note that c_i refers to the cost of the source configuration, and c_j refers to the cost of the target configuration. Three cases are possible: 1) A new configuration whose cost is higher than the maximal value c_{max} ¹ is automatically rejected. 2) A transition corresponding to a downhill move in the cost landscape ($c_j \leq c_i$) is always accepted. 3) An uphill move is accepted or rejected based on the probability $\exp(-(c_j - c_i) / T)$ that decreases exponentially with the cost variation $c_j - c_i$, in a way similar to the Metropolis criterion. In that case, the level of selectivity of the transition test is controlled by the adaptive parameter T , called *temperature* only by analogy to statistical physics. Low temperatures limit the expansion to gentle slopes of the cost landscape, and high temperatures enable to climb steep slopes. In T-RRT, the temperature is dynamically tuned during the search process, based on the current number of consecutive rejections $nFail$ and on the maximal number of consecutive rejections $nFail_{\text{max}}$. 1) After each accepted uphill move, T is decreased to avoid over-exploring high-cost regions. More precisely, T is divided by the temperature adjustment factor α , and $nFail$ is reset to 0. 2) After each rejected uphill transition, different actions are possible, depending on the value of $nFail$. If $nFail$ is less than

¹A value can be provided for c_{max} when high-cost regions of the space have to be forbidden.

Algorithm 5: transitionTest (\mathcal{G} , c_i , c_j)

input : the maximal cost c_{\max}
the current temperature T
the temperature adjustment factor α
the current number of consecutive rejections $nFail$
the maximum number of consecutive rejections $nFail_{\max}$

output: *true* if the transition is accepted, *false* otherwise

```

1 if  $c_j > c_{\max}$  then
2   | return False
3 else if  $c_j \leq c_i$  then
4   | return True
5 else if  $\text{rand}(0, 1) < \exp(-(c_j - c_i) / T)$  then
6   |  $T \leftarrow T / \alpha$ 
7   |  $nFail \leftarrow 0$ 
8   | return True
9 else
10  | if  $nFail > nFail_{\max}$  then
11  |   |  $T \leftarrow T \cdot \alpha$ 
12  |   |  $nFail \leftarrow 0$ 
13  |   else
14  |     |  $nFail \leftarrow nFail + 1$ 
15  |   return False

```

Algorithm 6: refinementControl (\mathcal{G} , q_{near} , q_{rand})

input : the extension step-size δ
the refinement ratio ρ

output: *true* if refinement is not too high, *false* otherwise

```

1 if  $\text{distance}(q_{\text{near}}, q_{\text{rand}}) < \delta$  and  $\text{nbRefinementNodes}(\mathcal{G}) > \rho \cdot \text{nbNodes}(\mathcal{G})$  then
2   | return False
3 else
4   | return True

```

its maximal value $nFail_{\max}$, the temperature remains unchanged, but $nFail$ is incremented by 1. If $nFail$ is greater than $nFail_{\max}$, T is increased to facilitate exploration and avoid being trapped in a local minimum. More precisely, T is multiplied by the temperature adjustment factor α , and $nFail$ is reset to 0. According to experiments reported in [83], a value of 2 seems to be a good choice for the α parameter. Furthermore, note that the adaptive tuning of the temperature allows T-RRT to automatically balance its bias toward low-cost regions with the Voronoi bias of RRT. This can be controlled by changing the value of the $nFail_{\max}$ parameter, which thus determines a trade-off between low computation time and high quality of the produced paths. A small value (e.g. 10) leads to a greedy search, and a greater value (e.g. 100) enables to produce higher-quality paths [83].

The adaptive tuning of the temperature ensures a given success rate for uphill transitions, but it can also produce an unwanted side-effect: T may be reduced by the acceptance of new states close to states already contained in the tree, whereas increasing T may be required to go over a local cost barrier and explore new regions of the space. Accepting such states only contributes to refining the exploration of low-cost areas already reached by the tree.

The objective of the `refinementControl` procedure (shown in Algorithm 6) is to limit this refinement and facilitate tree expansion toward unexplored regions of the space. The idea is to reject an expansion that would lead to more refinement if the number of refinement nodes already present in the tree is greater than a certain ratio ρ of the total number of nodes. A refinement node is defined as a node whose distance to its parent is less than the extension step-size δ . Another benefit of the refinement control is to limit the number of nodes in the tree, and thus to reduce the computational cost of the nearest-neighbor search. According to experiments performed in [83], a value of 0.1 seems to be a good choice for the ρ parameter. It has been shown that this refinement-control mechanism is beneficial mainly when dealing with low-dimensional spaces. However, its impact seems to be negligible on high-dimensional problems. Therefore, for simplicity’s sake, we do not include this procedure in the T-RRT-like algorithms we present in the rest of this thesis.

Theorem 3 (Probabilistic completeness of T-RRT). *In the space where configurations whose cost is greater than c_{\max} are regarded as part of $\mathcal{C}_{\text{obst}}$, the T-RRT algorithm is probabilistically complete [83].*

2.3 Optimal Path Planning

In some application contexts, beyond the computation of high-quality paths, it might be interesting to produce an optimal path, with respect to a given path quality criterion. This paradigm is referred to as “optimal path planning” [93]. This can mean, for instance, finding the shortest path, as is often done in robotics. However, in this thesis, we do not wish to restrict ourselves to such criteria that assess the quality of the path as a whole and ignore the costs of the configurations along the path. Our objective is to deal with a more general formulation of the optimal path planning problem, where the path-quality criterion is defined based on the configuration-cost function characterizing the cost space (as we describe later in this section). As examples, in robotics this kind of optimal path planning can result into looking for the path maximizing safety; in biology this means finding the motion minimizing the energy variation of a molecule.

When applied to the optimal path planning problem, classical grid-based methods, such as A* or D*, can compute resolution-optimal solution paths [148]. 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 shortest path, and the *Voronoi diagram* allows generating the path with optimal clearance [102]. Nevertheless, such methods are also limited to low-dimensional spaces, and can only deal with polygonal obstacles. On the other hand, classical sampling-based path planners can cope with high-dimensional spaces, but usually produce sub-optimal solutions because they focus on feasible path planning. As a complementary technique, after a solution path has been computed, it is common to improve the quality of this path during a post-processing phase involving so-called “smoothing” methods [69]. Nevertheless, such methods only allow to improve the path locally, and offer no guarantee of converging toward the global optimum. The first sampling-based path planner providing such guarantee was RRT* [90], that we present later in this section.

2.3.1 Theoretical Framework

Let $c_p : \Pi_{\text{free}} \rightarrow \mathbb{R}_+$ denote a quality criterion, associating to each collision-free path a positive cost value, and whose definition is based on the configuration-cost function $c : \mathcal{C} \rightarrow \mathbb{R}_+$. Note that c_p can be defined in different ways (as the integral of the cost, the mechanical work, the

maximal cost, etc) that we describe in the next section. The *optimal path planning problem* can be defined as follows:

Definition 4 (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 $c_p(\pi^*) = \min\{c_p(\pi) \mid \pi \in \Pi_{\text{feas}}\}$ if one exists, or report failure otherwise.*

Based on these notations, an optimal path planning problem is denoted by a quintuplet $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_p)$. 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 [93]. This concept is based on the notion of *weak clearance*. Given $\lambda \in \mathbb{R}_+$, a path $\pi \in \Pi_{\text{free}}$ is said to have weak λ -clearance if there exists a path $\pi' \in \Pi_{\text{free}}$ having strong λ -clearance and a homotopy $\psi : [0, 1] \rightarrow \Pi_{\text{free}}$ such that $\psi(0) = \pi$, $\psi(1) = \pi'$, and $\forall \alpha \in (0, 1)$, $\psi(\alpha)$ has strong λ_α -clearance for some $\lambda_\alpha \in \mathbb{R}_+$. Note that, if a path has strong λ -clearance, it obviously has weak λ -clearance. From that, robust optimality is defined as follows:

Definition 5 (Robust optimality). *An optimal solution path π^* to an optimal path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_p)$ is robustly optimal if it has weak λ -clearance and if any sequence of paths $\{\pi_m\}_{m \in \mathbb{N}}$ in Π_{free} such that $\lim_{m \rightarrow +\infty} \pi_m = \pi^*$ satisfies $\lim_{m \rightarrow +\infty} c_p(\pi_m) = c_p(\pi^*)$.*

In the context of optimal path planning, the evaluation of a planning algorithm should be based not only on the concept of probabilistic completeness, but also on the concept of *asymptotic optimality* [93]. A similar concept that has emerged in recent work, but that we do not consider in this thesis is *asymptotic near-optimality* [54, 118]. Asymptotic optimality can be interpreted as a notion of “almost-sure” convergence toward the optimal path, and has been defined as follows [93]:

Definition 6 (Asymptotic optimality). *An algorithm \mathcal{A} is asymptotically optimal if, for any optimal path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_p)$ 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.*

Note that it has been shown that RRT is not asymptotically optimal [93]. In other words, RRT cannot solve the optimal path planning problem. As a consequence, T-RRT is not asymptotically optimal either. Even though it yields high-quality paths when solving the cost-space path planning problem, T-RRT cannot solve the optimal path planning problem because it does not include any mechanism to improve its solution.

2.3.2 Path-Quality Criteria

The path-quality criterion $c_p : \Pi_{\text{free}} \rightarrow \mathbb{R}_+$ can be defined in several ways. As already mentioned, the first criterion to be utilized was path length. In the context of cost-space path planning, using path length would mean either 1) that the costs of configurations along the path are ignored or 2) that the configuration-cost function associate to every configuration the cost value 1. It is obviously more interesting to consider a path-quality criterion that takes a more complex configuration-cost function into account. We now present the various criteria that we consider in this thesis.

The most common criterion is the *integral of the cost* along a path. As a discrete approximation of the integral of the cost (IC) with constant step size $\delta = \frac{1}{n}$ (where n is the number of subdivisions of the path), the cost of a path π can be defined as

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

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 [83]. As a discrete approximation of the mechanical work (MW) with constant step size $\delta = \frac{1}{n}$, the cost of a path π can be defined as

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

Additionally, we consider other, simpler cost criteria to evaluate path quality, such as the maximal cost along the path (maxC), and the average cost (avgC). As discrete approximations with n subdivisions of the path π , these costs are defined as follows:

$$c_p(\pi) = \max \left\{ c \left(\pi \left(\frac{k}{n} \right) \right); k = 0..n \right\} \quad (\text{maxC})$$

$$c_p(\pi) = \frac{1}{n} \sum_{k=0}^n c \left(\pi \left(\frac{k}{n} \right) \right) . \quad (\text{avgC})$$

Which criterion is the most suited depends on the planning problem and on the characteristics of its expected optimal solution. Comparing quality criteria is out of the scope of this thesis. In our experiments, we use several of these criteria not to limit ourselves to a single criterion, which could bias the interpretation of the results. Finally, note that when using such quality criterion, as a slight abuse of language, we interchangeably utilize the expressions “high-quality path” and “low-cost path”.

2.3.3 The RRT* Algorithm

The RRT* algorithm was specifically proposed to solve the optimal path planning problem [93]. Indeed, it has been shown that RRT* offers asymptotic-optimality guarantees. This path planner has been successfully applied to various robotic problems [87, 93, 94] (some even involving kinodynamic planning [91] or manipulation tasks [131]) and to pursuit-evasion games [92]. However, RRT* has been most often used to optimize path length (without any configuration-cost function involved), and has only rarely been applied to more sophisticated quality criteria. In these rare cases, it has been evaluated only on cost-space path-planning problems involving coarse-grained, discrete configuration-cost functions. Furthermore, even though RRT* performs very well in low-dimensional spaces, it appears to converge slowly toward the optimum in high-dimensional spaces [49, 79].

RRT* was developed as an extension to RRT. The two algorithms differ mainly in the way connections are created between nodes of the tree. 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.

The pseudo-code of RRT* is shown in Algorithm 7. The part that differs from RRT concerns what is done after a new configuration q_{new} is built. First, a new node is created in \mathcal{T} to store q_{new} (line 7). Then, a search in \mathcal{T} 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). 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{T} . This dependency on n ensures that the radius decreases as \mathcal{T} grows. The next step consists of finding the configuration q_{par} in $Q_{\text{near}} \cup \{q_{\text{new}}\}$ to

Algorithm 7: RRT*

```

input : the optimal path planning problem  $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_p)$ 
         the dimension  $d$  of the  $\mathcal{C}$ -space
         the  $\gamma$  constant derived from the volume of  $\mathcal{C}_{\text{free}}$  [93]
output: the tree  $\mathcal{T}$ 
1  $\mathcal{T} \leftarrow \text{initTree}(q_{\text{init}})$ 
2 while not stoppingCriteria( $\mathcal{T}$ ) do
3    $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
4    $q_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}, q_{\text{rand}})$ 
5    $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}, q_{\text{rand}})$ 
6   if  $q_{\text{new}} \neq \text{null}$  then
7      $\text{addNewNode}(\mathcal{T}, q_{\text{new}})$ 
8      $n \leftarrow \text{numberOfNodes}(\mathcal{T})$ 
9      $Q_{\text{near}} \leftarrow \text{findNodesInBall}(\mathcal{T}, q_{\text{new}}, \gamma (\log(n) / n)^{1/d})$ 
10     $q_{\text{par}} \leftarrow \text{findParentMinimizingCostFromInit}(q_{\text{new}}, q_{\text{near}}, Q_{\text{near}}, c_p)$ 
11     $\text{addNewEdge}(\mathcal{T}, q_{\text{par}}, q_{\text{new}})$ 
12    foreach  $q_n \in Q_{\text{near}}$  do
13       $\pi \leftarrow \text{pathInSpace}(q_{\text{new}}, q_n)$ 
14      if  $\text{costFromInit}(q_{\text{new}}) + c_p(\pi) < \text{costFromInit}(q_n)$  and
15       $\text{isCollisionFree}(\pi)$  then
16         $\text{removeEdge}(\mathcal{T}, \text{parent}(q_n), q_n)$ 
17         $\text{addNewEdge}(\mathcal{T}, q_{\text{new}}, q_n)$ 
18 return  $\mathcal{T}$ 

```

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 maximal quality (i.e. minimal cost). This is done by computing, for all $q_n \in Q_{\text{near}} \cup \{q_{\text{near}}\}$, the cost $c_p(\pi_n^{\mathcal{T}}) + c_p(\pi_n^{\mathcal{C}})$, where $\pi_n^{\mathcal{T}}$ is the path between q_{init} and q_n in \mathcal{T} , and $\pi_n^{\mathcal{C}}$ is the path between q_n and q_{new} in \mathcal{C} . Finally, since the addition of a new node in \mathcal{T} potentially leads to the apparition of new paths having lower costs than those currently in \mathcal{T} , some rewiring might be performed (lines 12–16). For each $q_n \in Q_{\text{near}}$, 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{T} , q_{new} becomes the new parent of q_n in \mathcal{T} .

Theorem 4 (Probabilistic completeness of RRT*). *The RRT* algorithm is probabilistically complete [93].*

Let us assume that the γ constant involved in RRT* satisfies

$$\gamma > 2 \left(1 + \frac{1}{d}\right)^{\frac{1}{d}} \left(\frac{\mu(\mathcal{C}_{\text{free}})}{\zeta_d}\right)^{\frac{1}{d}}, \quad (2.1)$$

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 $\mu(\cdot)$ is an operator measuring volumes. Under this assumption, RRT* has proven to be asymptotically optimal [93]. The lower bound on γ expressed in (4.1) is the minimal value allowing RRT* to be asymptotically optimal. Keeping in mind that increasing the value of γ raises the computational cost of an RRT* iteration (because of the increased number of neighbors to consider), this lower bound represents the optimal tradeoff between efficiency and asymptotic optimality.

Theorem 5 (Asymptotic optimality of RRT*). *If the γ constant satisfies (4.1), the RRT* algorithm is asymptotically optimal [93].*

2.4 Parallel Path Planning

2.4.1 General Parallel Approaches

The idea of improving path planning performance using parallel computation has been explored for several decades. A survey of some early work suggests a classification scheme to review various path planning approaches and related parallel processing methods [74]. A more recent trend is to exploit the multi-core technology available on many of today’s PCs, which allows having multiple threads collaboratively solving a problem [40]. Another recent trend consists of using shared-memory models on many-core Graphics Processing Units (GPUs) [15, 17, 129, 130].

Among classical approaches, the *embarrassingly parallel paradigm* exploits the fact that some randomized algorithms, such as PRM, are what is termed “embarrassingly parallel” [6]. The massive inherent parallelism of the basic PRM algorithm allows reaching a significant speedup, even with simple parallelization strategies, especially on shared-memory architectures. In this approach, computation time is minimized by having several processes cooperatively building the road-map.

Another simple approach is known as the *OR parallel paradigm*. It was first applied to theorem proving, before providing a parallel formulation for the Randomized Path Planner (RPP) [27]. Its principle is to have several processes running the same sequential randomized algorithm, each one trying to build its own solution. The first process to reach a solution reports it and broadcasts a termination message. The idea is to minimize computing time by finding a small-sized solution. Despite its simplicity, this paradigm has been successfully applied to other randomized algorithms [26].

More sophisticated approaches are the *scheduler-processor scheme* and the *master-slave scheme* developed to distribute the computation of the Sampling-based Roadmap of Trees (SRT) algorithm [134, 135]. In a first step, several trees (called milestones, and that can be RRTs or ESTs) are computed in parallel by all processes. In a second step, a scheduler evenly distributes the computation of edges linking these trees among the other processes [134]. As an improvement to this, several masters can cooperate to distribute the workload among their respective slave processes [135]. More generally, an approach based on growing several independent trees, such as the Rapidly exploring Random Forest of Trees (RRFT) [9, 56] or RRTLocTrees [149], can lead to a straightforward parallelization.

2.4.2 Parallel RRT

Only little work relates to parallelizing RRT [2, 25, 44, 142]. The first approach applied the simple *OR parallel* and *embarrassingly parallel* paradigms, and a combination of both [25]. To benefit from the simplicity of the shared-memory model, the embarrassingly parallel algorithm is run on a single Symmetrical Multi-Processor (SMP) node of a multi-nodes parallel computer. The only communication involved is a termination message that is broadcast when a solution is reached, and some coordination is required to avoid concurrent modifications of the tree. This scheme does not make use of the full computational power of the parallel platform, contrary to the OR parallel algorithm, which is run on all processors of all nodes. The same paradigms are also applied on a dual-core CPU in [2], where they are renamed *OR* and *AND* implementations. In the Open Motion Planning Library (OMPL), the AND paradigm is implemented via multi-threading, and thus for shared memory². In [142], the OR paradigm is used on shared memory. A more recent and very successful approach for parallelizing RRT and RRT* on shared-memory architectures is presented in [78]. It is primarily based on lock-free shared data structures that are updated using atomic compare-and-swap (CAS) operations,

²http://ompl.kavrakilab.org/classompl_1_1geometric_1_1pRRT.html#gpRRT

and on a partition-based sampling (in other words, each thread is assigned a distinct region of the space).

In Chapter 5 of this thesis, we focus on the less studied issue of parallelizing RRT on distributed-memory architectures, using the Message Passing Interface (MPI). To the best of our knowledge, there has been only one attempt to develop a parallel version of RRT on distributed memory. In [44], the construction of the tree is distributed among several autonomous agents, using a message passing model. However, no explanation is given on how the computation is distributed, and how the tree is reconstructed from the parts built by the agents.

The objective of our work was to provide parallel versions of the basic (single-tree) RRT algorithm. This work was not about parallelizing subroutines of RRT, such as is done for collision detection in [15], nor about parallelizing specific variants of RRT, such as is done for the *anytime* RRT in [127]. In Chapter 5 we present three parallel versions of RRT based on classical parallelization schemes: OR parallel RRT, Distributed RRT and Manager-worker RRT [48, 50]. Since this work was published, two extensions of our Distributed RRT have been proposed [81].

Chapter 3

Efficient Cost-based Path Planning in Complex Continuous Cost Spaces

In path planning, instead of aiming only for collision avoidance, it can be important to compute high-quality paths. In recent years, variants of classical sampling-based algorithms have been developed to take cost functions into account during the exploration of the configuration space. Among them, the Transition-based RRT (T-RRT) algorithm was developed as an extension of RRT specifically targeting cost-space path planning (cf. Section 2.2).

T-RRT has been applied to diverse problems in robotics [12, 79, 83] (some even involving human–robot interactions [114]) and computational structural biology [79, 82]. These problems are challenging because they involve high-dimensional spaces and complex continuous cost functions. Our objective is to develop extensions of T-RRT that improve its performance and allow solving even more difficult problems, as well as new kinds of problems. Since T-RRT was devised as a mono-directional algorithm similar to the *Extend* RRT, there is room for improvement, based on ideas that have proven beneficial for RRT, and new ideas.

In this chapter, we introduce several extensions of T-RRT, starting from enhancements of its original mono-directional variant, then a bidirectional variant, and finally a multiple-tree variant. First, we improve the performance of the mono-directional T-RRT by modifying the implementation of its transition test. We also show that using a *Connect* T-RRT or a *Goal-biased* T-RRT can sometimes be beneficial. Second, we present a bidirectional extension of T-RRT that reduces running time and sometimes increases (or otherwise maintains) path quality. We also show that the *Bidirectional* T-RRT can produce better-quality paths than RRT* in high-dimensional spaces. Third, we propose a multi-tree extension of T-RRT that can compute a path visiting a set of waypoints, and we show that it outperforms path planners involving the Bidirectional T-RRT. We also provide a preliminary discussion on an anytime version of the *Multi-T-RRT* that is detailed later in this thesis (cf. Chapters 6 and 7). Finally, note that all the T-RRT variants presented in this chapter have been implemented and evaluated within the path-planning platform *Move3D* [145].

3.1 Extensions of the Mono-directional Variant of T-RRT

In this section, we present various extensions of the original version of T-RRT, that was proposed as a mono-directional algorithm similar to the *Extend* RRT (cf. Section 2.1). Due to this similarity, one may think that the modifications improving the mono-directional RRT are also beneficial to the mono-directional T-RRT. We show that this is not always true for the *Connect* and *Goal-biased* variants of T-RRT. On the other hand, we propose some enhancements of T-RRT’s transition test that improve performance. Before that, we start by presenting the cost-space path-planning problems used in our evaluation, as they intervene in all the experiments reported in this section and in Section 3.2.

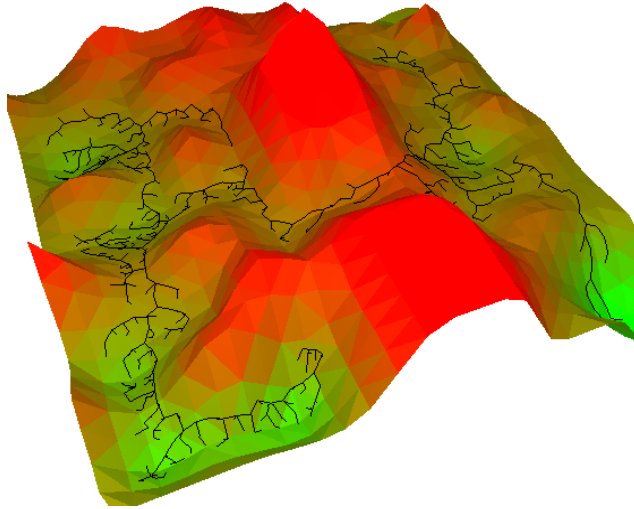


Figure 3.1: Search tree built by T-RRT on the *Mountains* problem. On this 2D cost-map, the cost is color-coded (from green to red) and represented by the elevation.

3.1.1 Path Planning Problems and Evaluation Settings

Throughout this section and Section 3.2, we use the same four cost-space path-planning problems to evaluate the performance of the mono-directional and bidirectional variants of T-RRT we propose. In Section 3.3, we use similar but larger-scale problems that are better suited to evaluate the multi-tree variants of T-RRT.

The examples we use in this section differ in terms of geometrical complexity, configuration-space dimensionality and cost-function type. The *Mountains* problem is the 2D cost-map illustrated by Fig. 3.1, in which the cost is represented by the elevation. The *Stones-small* problem (presented in Fig. 3.2) is a 2-degrees-of-freedom (DoF) problem in which a disk has to go across a space cluttered with rectangular-shaped stones. The objective being to maximize clearance, the cost of a configuration is the inverse of the distance between the disk and the closest obstacle. The *Manipulator* problem (illustrated in Fig. 3.3) involves a 6-DoF manipulator arm that has to get a stick through a hole in a wall while maximizing the clearance of the stick. Therefore, the cost of a configuration is the inverse of the distance between the stick and the obstacles. Finally, in the *Engine-simple* problem (shown in Fig. 3.4) the same robotic arm holds a sensor with a spherical extremity used to inspect a car engine. The objective being to keep the sensor as close as possible to the engine’s surface, the cost of a configuration is the distance (in millimeters) between the sphere and the engine. In this example, we set the cost-threshold value c_{\max} (cf. Section 2.2.2) to 100 because of the sensor’s range. The other examples do not require the use of a cost threshold, therefore we set $c_{\max} = \infty$. Also, following [83], in all these examples, we initialize the temperature T (cf. Section 2.2.2) to 10^{-6} .

To fairly assess the benefits of each T-RRT variant, the solution paths are not subjected to a post-processing phase involving “smoothing” methods [69]. On a given problem, we evaluate each algorithm on the basis of the running time t (in seconds), the number of expansion attempts X , the number of nodes N in the produced tree, and various quality criteria applied to the extracted path: the average cost $avgC$, the maximal cost $maxC$, the mechanical work MW , and the integral of the cost IC (cf. Section 2.3.2). For all variables, we give values averaged over 100 runs. Results were obtained on an Intel Core i5 processor at 2.6 GHz with 8 GB of memory.

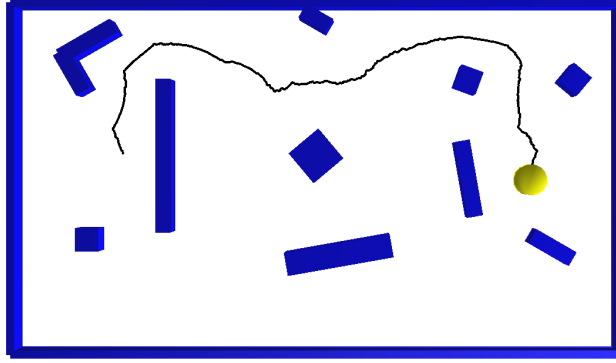


Figure 3.2: Path computed by T-RRT on the *Stones-small* problem. The cost is the inverse of the distance between the 2-DoF yellow disk and the blue rectangular-shaped obstacles.

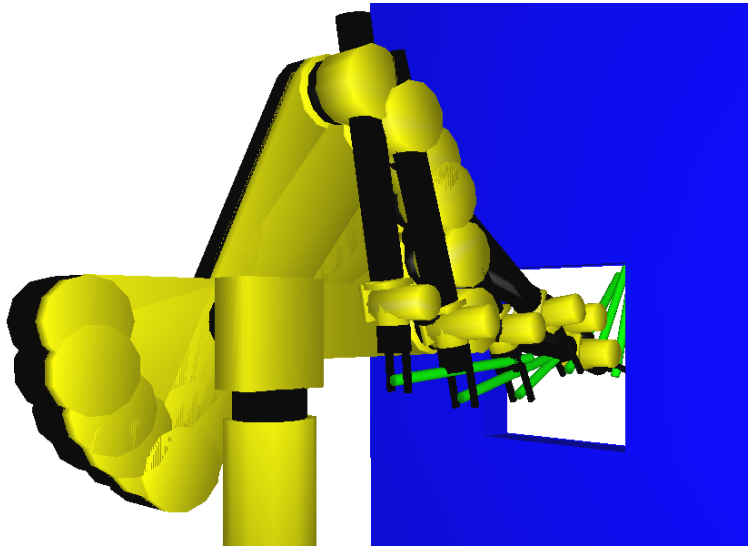


Figure 3.3: Trace of a path computed by T-RRT on the *Manipulator* problem. A 6-DoF manipulator arm has to get a stick through a hole while maximizing its clearance (i.e. its distance to the obstacles).

3.1.2 Improvement of the Transition Test

Compared to its original version (cf. Algorithm 5 in Section 2.2.2) [83], the version of the transition test we propose (shown in Algorithm 8) includes three enhancements. They not only affect the way the Metropolis-like test is performed, but also the way the temperature is adjusted during the exploration. Because this is not very informative, we do not report here the numerical evaluation that we have performed to assess the impact of these modifications. Results we have obtained show that they improve the performance of T-RRT: running times are significantly reduced without incurring any loss in path quality.

The first enhancement appears at line 5. We have replaced the boolean expression $\text{rand}(0,1) < \exp(-(c_j - c_i)/T)$ by $\exp(-(c_j - c_i)/T) > p$ with $p = 0.5$ to better control the stochastic aspect of the Metropolis-like test. Using $\text{rand}(0,1)$ instead of a fixed probability p has the following detrimental consequence: if $\text{rand}(0,1)$ is close to 1, any move has high chances to be accepted, even an undesirable steep uphill move; if $\text{rand}(0,1)$ is close to 0, any move is likely to be rejected, even a gentle uphill move. We have varied the value of p and observed that this change has no impact on the exploration, except if p is close to 1. In fact, the adaptive nature of the temperature compensates any change in p : if p is lowered, the temperature simply reaches higher values.

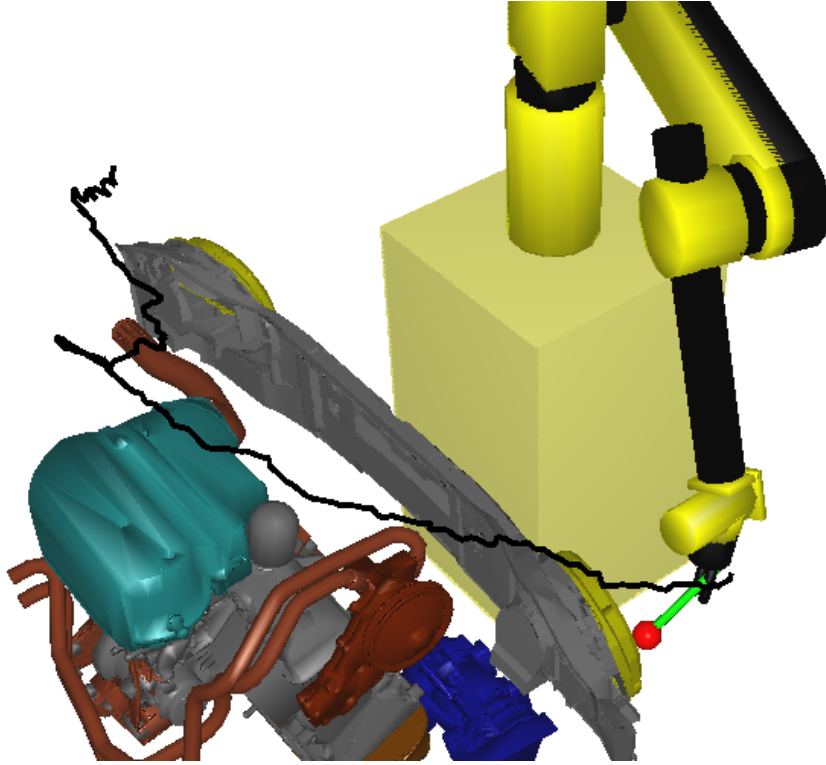


Figure 3.4: Path computed by T-RRT on the *Engine-simple* problem. The 6-DoF manipulator arm holds a sensor (the red sphere) that has to follow the surface of the car engine while remaining as close to it as possible.

Algorithm 8: transitionTest (\mathcal{G}, c_i, c_j)

input : the cost threshold c_{\max}
the current temperature T
the temperature increase rate T_{rate}
output: *true* if the transition is accepted, *false* otherwise

- 1 **if** $c_j > c_{\max}$ **then**
- 2 | **return** False
- 3 **else if** $c_j \leq c_i$ **then**
- 4 | **return** True
- 5 **else if** $\exp(-(c_j - c_i) / T) > 0.5$ **then**
- 6 | $T \leftarrow T / 2^{(c_j - c_i) / (0.1 \cdot \text{costRange}(\mathcal{G}))}$
- 7 | **return** True
- 8 **else**
- 9 | $T \leftarrow T \cdot 2^{T_{\text{rate}}}$
- 10 | **return** False

The second improvement (line 9) consists of progressively increasing the temperature after each rejection, instead of increasing it by performing a single larger jump after a given number of consecutive rejections. For that, after each rejected uphill transition, T is multiplied by $2^{T_{\text{rate}}}$, where $T_{\text{rate}} \in (0, 1]$ is the temperature increase rate. The T_{rate} parameter determines a trade-off between low computation time and high quality of the produced paths. A value not too small (e.g. 0.1) leads to a greedy search, and a lower value (e.g. 0.01) enables to produce higher-quality paths. In the sequel, we use only these two values for T_{rate} .

The third enhancement appears at line 6 and is borrowed from [79]. It consists of providing an implicit refinement control mechanism by making the temperature reduction dependent on the cost variation associated with an accepted uphill transition. To achieve that, T is divided by 2 to the power of $(c_j - c_i) / (0.1 \cdot \text{costRange}(\mathcal{G}))$, where $\text{costRange}(\mathcal{G})$ is the cost difference between the highest-cost and the lowest-cost configurations among those stored in the nodes of the graph \mathcal{G} .

3.1.3 Connect and Goal-biased Extensions of T-RRT

Compared to the basic *Extend* RRT, several classical RRT extensions are known to improve performance [106]. For example, the *Goal-biased* RRT may converge faster toward the goal. Also, with the *Connect* RRT, the search tree generally grows faster. However, when it comes to T-RRT, improving performance means not only reducing running time, but also increasing (or at least maintaining) path quality. Thus, one may wonder whether applying these modifications of RRT to T-RRT is beneficial.

Goal-biased T-RRT

In the same way as it is done for RRT, implementing the Goal-biased T-RRT consists of modifying the `sampleRandomConfiguration` function (line 3 in Algorithm 4, Section 2.2.2) so that it returns q_{goal} with a probability goalBias . Results of the experimental evaluation of the Goal-biased T-RRT (with $\text{goalBias} = 0.01$ and 0.1) are shown in Table 3.1. When compared to the Extend T-RRT, the Goal-biased T-RRT yields lower running times on all examples. However, when $\text{goalBias} = 0.01$, path quality improves on all problems if $T_{\text{rate}} = 0.1$, but not if $T_{\text{rate}} = 0.01$. On the contrary, when $\text{goalBias} = 0.1$, path quality globally improves if $T_{\text{rate}} = 0.01$, but not if $T_{\text{rate}} = 0.1$, especially on 2-DoF problems. Therefore, the Goal-biased extension of T-RRT can be beneficial, but it lacks robustness with respect to path-quality improvement. We show in the next section that, in the context of a simple “init-to-goal” problem, the bidirectional variant of T-RRT is a better choice than the Goal-biased T-RRT.

Connect T-RRT

Contrary to the Connect RRT, the Connect T-RRT can be implemented in various ways, due to the presence of the transition test. The simplest way consists of iterating the `extend` and `transitionTest` functions (lines 6 and 7 in Algorithm 4) as long as there is no collision and the transition test is passed (without adding the intermediate states to the tree). When compared to other variants (delaying temperature tuning, or limiting uphill transitions), this implementation yields the best results, which are reported in Table 3.1. In order not to overload the tables, results obtained with the other Connect T-RRT variants are not reported here. Results in Table 3.1 show that, in comparison to using the Extend T-RRT, using the Connect T-RRT reduces running time, but does not always increase path quality, especially if $T_{\text{rate}} = 0.01$. Therefore, if achieving high path-quality is more important than reducing running time, the Extend T-RRT should be preferred over the Connect T-RRT.

3.2 Bidirectional Extension of T-RRT

The *Bidirectional* RRT is known to be more efficient than the mono-directional RRT [106]. Therefore, in this section, we analyze whether it is possible to develop a Bidirectional version of T-RRT that improves its performance. As already mentioned, this means not only reducing running time, but also increasing (or at least maintaining) path quality. We compare several possible implementations for the components involved in a Bidirectional T-RRT and select the best ones; this concerns mainly how trees should be expanded and linked. Then, we analyze

Table 3.1: Evaluation on four cost-space path-planning problems of several variants of T-RRT: 1) Extend T-RRT, 2) Goal-biased T-RRT with $goalBias = 0.01$, 3) Goal-biased T-RRT with $goalBias = 0.1$, and 4) Connect T-RRT. All values are averaged over 100 runs.

$T_{rate} = 0.1$		$avgC$	$maxC$	MW	IC	t (s)	N	X
<i>Mountains</i>	1	17.4	24.8	29.3	323	2.2	1,660	6,260
	2	17.4	24.6	28.6	315	0.4	778	2,280
	3	17.7	25.7	28.6	303	0.1	433	1,400
	4	16.4	23.2	31.2	378	0.4	633	2,530
<i>Stones-small</i>	1	3.17	6.39	15.9	322	0.6	652	4,080
	2	3.17	6.36	15.2	312	0.5	594	3,520
	3	3.22	6.75	15.9	312	0.3	512	2,910
	4	3.15	6.14	13.0	333	0.3	399	3,170
<i>Manipulator</i>	1	6.3	8.0	8.0	1,020	2.1	776	7,130
	2	6.1	7.5	4.4	831	0.3	201	2,060
	3	6.2	7.6	3.3	742	0.1	92	921
	4	5.9	7.9	8.2	1,160	1.0	492	3,910
<i>Engine-simple</i>	1	24.1	80.9	379	8,900	11.3	321	2,720
	2	21.5	72.6	356	7,780	10.7	293	2,520
	3	19.1	73.3	324	6,590	9.9	270	2,340
	4	12.3	49.9	236	4,500	9.8	146	2,190
$T_{rate} = 0.01$		$avgC$	$maxC$	MW	IC	t (s)	N	X
<i>Mountains</i>	1	16.7	22.8	26.5	378	2.7	1,150	16,400
	2	16.7	22.8	26.2	380	1.6	885	12,600
	3	16.7	22.7	25.2	363	1.3	812	11,900
	4	16.5	22.8	29.2	427	1.4	749	14,700
<i>Stones-small</i>	1	2.95	5.81	11.7	289	5.2	711	36,000
	2	2.96	5.82	11.5	284	5.0	702	35,600
	3	2.98	5.80	11.1	276	4.7	683	34,900
	4	3.08	5.89	10.4	315	4.6	597	39,000
<i>Manipulator</i>	1	5.8	6.6	3.8	911	6.9	709	26,400
	2	5.8	6.6	2.5	759	0.8	119	6,820
	3	5.8	6.7	2.3	719	0.6	89	5,640
	4	5.5	7.2	5.3	1,060	2.9	415	15,300
<i>Engine-simple</i>	1	1.7	10.9	88.9	626	78	288	23,600
	2	1.8	11.0	91.3	658	78	293	23,500
	3	1.6	9.9	87.4	587	77	274	23,300
	4	1.8	12.6	78.3	642	64	202	20,600

cost profiles of paths produced by the Bidirectional T-RRT and the Extend T-RRT, to better understand why the former sometimes outperforms the latter with respect to path quality. We also compare these two variants of T-RRT to RRT*, and show that T-RRT produces paths of higher quality than RRT* does in high-dimensional spaces. All these evaluations involve the settings and the cost-space path-planning problems used in Section 3.1. Finally, we present an industrial inspection problem featuring an aerial robot, that the Bidirectional T-RRT can solve efficiently, but not the Extend T-RRT.

Algorithm 9: Bidirectional T-RRT

```

input : the cost-space path planning problem  $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c)$ 
output: the tree  $\mathcal{T}$ 
1  $\mathcal{T}_1 \leftarrow \text{initTree}(q_{\text{init}})$ 
2  $\mathcal{T}_2 \leftarrow \text{initTree}(q_{\text{goal}})$ 
3 while not stoppingCriteria( $\mathcal{T}_1, \mathcal{T}_2$ ) do
4    $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
5    $q_{\text{near}}^1 \leftarrow \text{findNearestNeighbor}(\mathcal{T}_1, q_{\text{rand}})$ 
6    $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}^1, q_{\text{rand}})$ 
7   if  $q_{\text{new}} \neq \text{null}$  and transitionTest( $\mathcal{T}_1, c(q_{\text{near}}^1), c(q_{\text{new}})$ ) then
8     addNewNode( $\mathcal{T}_1, q_{\text{new}}$ )
9     addNewEdge( $\mathcal{T}_1, q_{\text{near}}^1, q_{\text{new}}$ )
10     $q_{\text{near}}^2 \leftarrow \text{findNearestNeighbor}(\mathcal{T}_2, q_{\text{new}})$ 
11     $\mathcal{T} \leftarrow \text{attemptLink}(\mathcal{T}_1, q_{\text{new}}, \mathcal{T}_2, q_{\text{near}}^2, 2)$ 
12  swap( $\mathcal{T}_1, \mathcal{T}_2$ )
13 return  $\mathcal{T}$ 

```

3.2.1 The Bidirectional T-RRT Algorithm

In the most efficient implementation of the Bidirectional RRT, computation is divided between growing two trees (from q_{init} and q_{goal} respectively) and trying to connect them. At each iteration, an expansion is attempted from one tree toward a random configuration and, if it succeeds, an expansion is attempted from the other tree toward the new node, potentially leading to the junction of both trees; then, the roles of the trees are reversed by swapping them. We now explain why, due to cost considerations, one has to be careful when applying this scheme to T-RRT in order to improve its performance, and we present an efficient implementation of the Bidirectional T-RRT. As this is not very informative, we do not report here the performance results obtained with all possible variants of the Bidirectional T-RRT, and limit ourselves to the best one.

Tree Expansion

In the Bidirectional RRT, the attempt to expand one tree toward a random configuration can be done with an Extend or Connect function [106]. Which one works best depends on the problem at hand. The same happens for the Bidirectional T-RRT: using a Connect function leads to lower running times, except on the *Engine-simple* problem. However, even when the Connect function is computationally beneficial, this generally happens at the expense of path quality. Therefore, it seems preferable to expand the trees using an Extend function.

Tree Junction

In the Bidirectional RRT, the attempt to link both trees can use an Extend or a Connect function [106]. Employing the Extend function lacks efficiency because this requires the trees to come at a distance smaller than the extension step-size. This may lead the two trees to explore a wider space area than a single tree would do, which is indeed what we observe with the Bidirectional T-RRT on very cost-constrained problems, such as *Engine-simple*. The Bidirectional T-RRT is then slower than its mono-directional counterpart. Utilizing the Connect function is more efficient at reducing running time, but this happens again at the expense of path quality. The best junction strategy appears to be some kind of Connect function that creates no node, and only tries to add a linking edge. Also, this function should be applied only if the trees are closer than a given distance threshold, not to waste time

Algorithm 10: attemptLink($\mathcal{T}_1, q_1, \mathcal{T}_2, q_2, n$)

```

input : the extension step-size  $\delta$ 
output: the tree  $\mathcal{T}$ 
1  $\mathcal{T} \leftarrow \text{null}$ 
2 if distance( $q_1, q_2$ ) <  $10 \cdot \delta$  and isCollisionFree(path( $q_1, q_2$ )) then
3    $q_{\text{cur}} \leftarrow q_1$ 
4    $q_{\text{next}} \leftarrow \text{extend}(q_1, q_2)$ 
5   while  $q_{\text{next}} \neq \text{null}$  and transitionTest( $\mathcal{T}_1, c(q_{\text{next}}), c(q_{\text{cur}})$ ) do
6      $q_{\text{cur}} \leftarrow q_{\text{next}}$ 
7      $q_{\text{next}} \leftarrow \text{extend}(q_{\text{cur}}, q_2)$ 
8   if  $q_{\text{cur}} = q_2$  then
9      $\mathcal{T} \leftarrow \text{linkAndMerge}(\mathcal{T}_1, q_1, \mathcal{T}_2, q_2)$ 
10     $n \leftarrow n - 1$ 
11 return  $\mathcal{T}$ 

```

checking potential edges that are unlikely to be valid. If this threshold is too low, though, the tree junction becomes difficult, as with the Extend function. A value of ten times the extension step-size seems to achieve the right balance. Finally, we have observed that, depending on the planning problem and on the cost function, it can be more efficient to first check for the absence of collisions and then apply the transition test, or reciprocally. For sake of simplicity, we do not make this distinction in the algorithm presented here.

Bidirectional T-RRT

To sum up, the best implementation for the Bidirectional T-RRT is the one presented in Algorithm 9. At each iteration, one tree is expanded toward a random configuration. If a new node is created and passes the transition test, a connection to its nearest neighbor in the other tree is attempted, via the attemptLink procedure shown in Algorithm 10. This procedure starts by checking whether both nodes are closer than ten times the extension step-size, and whether the path between them is collision-free. If this is the case, successive RRT expansion steps toward the target node are performed (without creating additional nodes), starting from the source node. If the transition test accepts all the expansion steps, the two nodes are connected by an edge, the two trees are merged into one, and the number of trees is decreased by 1. Note that we present a general implementation of this procedure because it is also used in the multiple-tree extension of T-RRT (cf. Section 3.3).

3.2.2 Evaluation and Analysis

Performance results obtained with the Bidirectional T-RRT on the four cost-space path-planning problems presented in Section 3.1 are reported in Table 3.2. Compared to using the mono-directional T-RRT, using the Bidirectional T-RRT greatly reduces running time, up to an order of magnitude. Moreover, this globally improves path quality: all path-cost measurements are reduced, sometimes very significantly (e.g. on the *Manipulator* problem), apart from a few exceptions (mainly on the *Mountains* problem) for which we observe a small increase. Our Bidirectional scheme thus significantly improves the performance of T-RRT. Furthermore, if we compare the results reported in Tables 3.1 and 3.2, we observe that the Bidirectional T-RRT does not always outperform the Goal-biased and Connect T-RRT, but it is more robust when it comes to improving path quality.

Table 3.2: Evaluation of the Bidirectional T-RRT on four cost-space path-planning problems, in comparison to the mono-directional T-RRT. All values are averaged over 100 runs.

$T_{\text{rate}} = 0.1$		$avgC$	$maxC$	MW	IC	t (s)	N	X
<i>Mountains</i>	mono	17.4	24.8	29.3	323	2.2	1,660	6,260
	bi	17.7	23.9	30.5	330	0.1	282	982
<i>Stones-small</i>	mono	3.17	6.39	15.9	322	0.6	652	4,080
	bi	3.15	6.35	14.6	313	0.1	251	1,790
<i>Manipulator</i>	mono	6.3	8.0	8.0	1,020	2.1	776	7,130
	bi	6.0	7.6	3.5	791	0.1	101	1,170
<i>Engine-simple</i>	mono	24.1	80.9	379	8,900	11.3	321	2,720
	bi	21.6	74.5	332	7,520	8.7	254	2,530

$T_{\text{rate}} = 0.01$		$avgC$	$maxC$	MW	IC	t (s)	N	X
<i>Mountains</i>	mono	16.7	22.8	26.5	378	2.7	1,150	16,400
	bi	16.7	22.8	27.2	370	0.9	861	11,700
<i>Stones-small</i>	mono	2.95	5.81	11.7	289	5.2	711	36,000
	bi	2.86	5.72	10.7	282	1.6	548	23,400
<i>Manipulator</i>	mono	5.8	6.6	3.8	911	6.9	709	26,400
	bi	5.7	6.7	2.3	771	0.8	112	7,410
<i>Engine-simple</i>	mono	1.7	10.9	88.9	626	78	288	23,600
	bi	1.6	7.5	87.2	575	69	304	25,200

Analysis of Cost Profiles

The cost profiles of paths obtained on the *Manipulator* problem reveal how the Bidirectional T-RRT can improve path quality (cf. Fig. 3.5). In this specific situation, both tree roots are in low-cost areas, and the solution path has to go through a saddle of the cost landscape (that is located approximately mid-way between q_{init} and q_{goal}). When T-RRT starts a descending phase after crossing the saddle, the temperature is high because of the previous ascension, making the transition test less selective: some uphill moves are accepted on the way down. The produced path is then a succession of downhill and uphill steps, leading to a jerky cost profile. On the contrary, when T-RRT is on an ascending phase, it is hard to go up: few uphill moves are accepted from a given node. But, these moves enable to reach the saddle and will eventually appear in the extracted path. An ascending path is thus a rather smooth succession of uphill moves, as can be seen in Fig. 3.5 for the first half of both cost profiles. Furthermore, with the Bidirectional T-RRT, the second half of the path is also built during an ascending phase, performed by the second tree. Taken in reverse direction, it appears as a smooth descent. Note that what is observed in this specific situation might not be generalizable.

Tree Growth Bias

Besides its tree-linking role, the junction procedure proposed in [106] introduces a bias in the search process: at each iteration, one tree is potentially grown toward a new node from the other tree. While using the tree-linking mechanism presented in Algorithm 10, we evaluate the impact of this bias. For that, at each iteration, if a new node is created in the tree \mathcal{T}_a , and if the junction to \mathcal{T}_b fails, we try growing \mathcal{T}_b toward the new node in \mathcal{T}_a using an Extend function. After evaluation, this bias appears to increase running time on some problems and to globally decrease path quality. We therefore reject it.

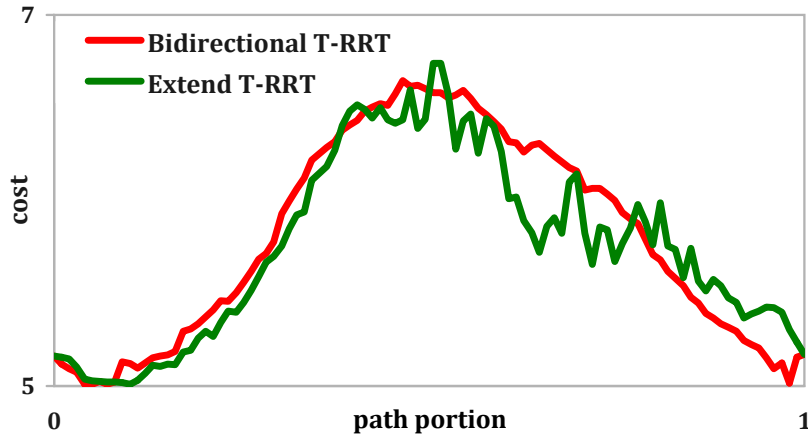


Figure 3.5: Cost profiles of two paths produced on the *Manipulator* problem by the Extend and Bidirectional T-RRT respectively. These paths are representative in the sense that their associated cost measurements are close to average values obtained over 100 runs.

Common Expansion Direction

We also evaluate a variant of the Bidirectional T-RRT directly adapted from a variant of the Bidirectional RRT [106]. In this variant, at each iteration, both trees are grown toward the same random configuration, and up to two junctions are attempted depending on the number of new nodes. This variant appears to be less efficient than the one presented in Algorithm 9. Its running time is slightly higher because of the greater number of attempted junctions. But more importantly, the quality of the paths produced is globally reduced. Therefore, we do not retain this modification.

Balanced Trees and Local Temperature

Finally, we evaluate two other versions of the Bidirectional T-RRT. The first one, which has proven to be beneficial to RRT on some problems, consists of ensuring that both trees remain balanced (in terms of number of nodes). The second one is specific to T-RRT and involves having a separate temperature associated to each tree. After evaluation, it is unclear whether these modifications are advantageous or not. They both appear to have sometimes a positive impact and sometimes a negative impact on performance.

Comparison with RRT*

We now compare the Extend and Bidirectional T-RRT to RRT*. On the RRT* side, the path-quality criterion to be optimized is chosen as being the mechanical work of a path, in order to ensure a fair comparison with T-RRT. Indeed, even though T-RRT is not conceived to be optimizing any quality criterion, it naturally tends to minimize the mechanical work of a path [83]. To simulate an optimization process on the T-RRT side, we vary the value of the T_{rate} parameter: decreasing T_{rate} increases running times, but improves the quality of solution paths. In this experiment, we run the T-RRT variants with $T_{\text{rate}} = 0.1$ and $T_{\text{rate}} = 0.01$. Results are reported in Fig. 3.6. We can see that, on the *Mountains* problem, RRT* quickly finds a better solution than T-RRT, even though the Bidirectional T-RRT performs equally well for 0.1 s. On the *Stones-small* problem, RRT* cannot find a solution in less than 0.5 s, contrary to the Bidirectional T-RRT, that succeeds in 0.1 s; but, given enough time, RRT* converges toward a better solution. On the *Manipulator* and *Engine-simple* problems (involving a 6-DoF manipulator arm), even in its Extend form, T-RRT outperforms RRT*. As pointed out in [79], RRT* converges slowly on high-dimensional problems.

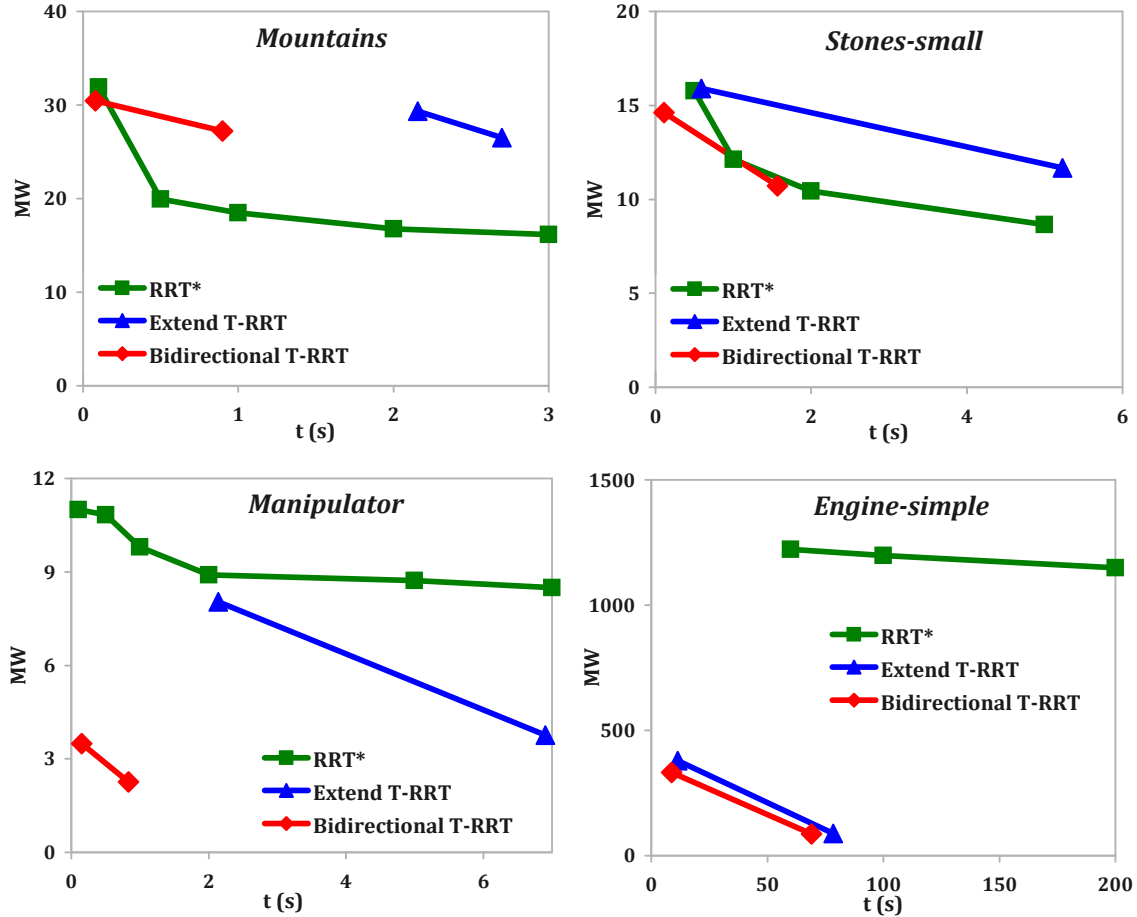


Figure 3.6: Path quality (measured by the mechanical work MW) in relation to running time (t , in seconds) for solution paths produced by T-RRT and RRT* on four cost-space path-planning problems. For each segment representing T-RRT, the left point corresponds to $T_{\text{rate}} = 0.1$ and the right point to $T_{\text{rate}} = 0.01$. Values are averaged over 100 runs.

3.2.3 Industrial Inspection Problem

This section presents a cost-space path-planning problem for a flying robot in a dense industrial environment, as illustrated by Fig. 3.7. For safety reasons, the quadrotor has to move in this environment while maximizing its distance to obstacles. This scenario is an example of industrial inspection involving aerial robots, such as those addressed in the framework of the ARCAS project. One of the goals of this project is to develop robot systems for the inspection of industrial installations that are difficult to access for humans.

In this example, the quadrotor is modeled as a 3-DoF sphere (i.e. a free-flying object) representing the safety zone around the robot; therefore, no visibility constraint is considered. We assume that the motions of the quadrotor are performed quasi-statically, thus neglecting dynamic constraints. We restrict the problem to planning in position, controllability issues being out of the scope of this thesis.

When running the Extend T-RRT on this problem, we observe that only 38 of the 100 runs succeed in less than five minutes with $T_{\text{rate}} = 0.1$, and 67 with $T_{\text{rate}} = 0.01$. On the other hand, the Bidirectional T-RRT can find a solution in less than 3 s when $T_{\text{rate}} = 0.1$, and in about 38 s when $T_{\text{rate}} = 0.01$ (on average over 100 runs). The example trajectory in Fig. 3.7 is typical of what the Bidirectional T-RRT produces when $T_{\text{rate}} = 0.01$: it shows that the quadrotor follows a convoluted path in order to maximize clearance. When $T_{\text{rate}} = 0.1$, solution paths are more diverse, some being shorter but having a lower clearance.

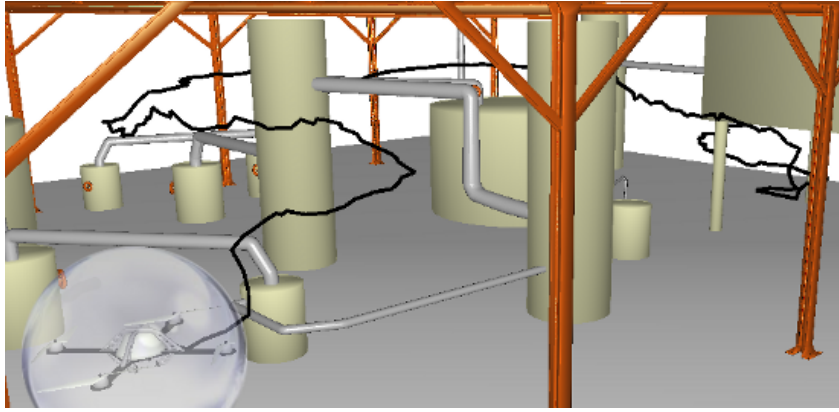


Figure 3.7: Path produced by the Bidirectional T-RRT for a quadrotor flying in a dense industrial environment.

3.3 Multiple-Tree Extension of T-RRT

In this section, we propose a multiple-tree variant of T-RRT, named *Multi-T-RRT*. As multiple-tree approaches to sampling-based path planning have been developed for various applications, with different objectives in mind, we start by specifying the scope of our approach within this general context. In short, our aim is to develop a single-query planner that can compute a path going through a given set of waypoints in a fixed environment. Since there exist numerous multi-tree path planners that involve RRT, we evaluate existing strategies used to expand and connect trees, and we select the most efficient ones to develop the Multi-T-RRT. In addition, this requires dealing with the subtleties related to the presence of cost constraints. Then, we evaluate the Multi-T-RRT and show that it outperforms path planning schemes involving the Bidirectional T-RRT. Finally, we propose a useful-cycle addition procedure that leads to an anytime version of the Multi-T-RRT, allowing for a continual improvement of the solution path. We also show that this variant of the Multi-T-RRT yields a better convergence rate for solution-path quality than PRM and $\text{RRT}_{\text{obst way}}$. Only preliminary concepts and results are presented in this section; the anytime variant of T-RRT will be discussed more thoroughly in Chapter 4.

3.3.1 Scope of the Approach

To develop the Multi-T-RRT, we have surveyed several techniques proposed in similar work on multi-tree approaches to sampling-based path planning. Some approaches aim at solving single-query problems, the way RRT usually works, but involve the construction of several RRTs to reach a solution [9, 34, 56, 57, 63, 149, 157, 158]. Others are multiple-query approaches similar to PRM, where RRT is used as a local planner [108, 122, 134]. Others focus on dynamic environments and build several RRTs at different points in time [67, 166]. The version of the Multi-T-RRT we present here is a single-query planner building several T-RRTs to find a path. We do not deal with multiple queries or dynamic environments.

Growing multiple trees on the configuration space can be done in various ways. The aim can be to have several RRTs rooted in different regions of the space to ensure a broader exploration [34, 57, 63, 149, 157, 158]. In this context, trees are initialized and grown independently of one another. Other approaches aim at maintaining a road-map of RRTs over the space [67, 108, 134, 166]. In this case, trees can be created or modified as a result of merging, splitting or pruning operations. Other approaches make use of sub-trees produced by previous queries [108, 166]. Others build RRTs in different subspaces, independently of each other [9, 56]. Finally, RRTs can be reduced to local connections between components of a large road-map [122]. In this work, we focus on growing several T-RRTs rooted at given waypoints.

Algorithm 11: Multi-T-RRT

```

input : the cost-space path planning problem  $(\mathcal{C}, \{q_{\text{init}}^k \mid k = 1..n\}, c)$ 
         where  $q_{\text{init}}^k, k = 1..n$  are a set of waypoints
output: the tree  $\mathcal{T}$ 
1 for  $k = 1..n$  do
2    $\mathcal{T}_k \leftarrow \text{initTree}(q_{\text{init}}^k)$ 
3 while not  $\text{stoppingCriteria}(\{\mathcal{T}_k \mid k = 1..n\})$  do
4    $\mathcal{T}' \leftarrow \text{chooseNextTreeToExpand}()$ 
5    $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
6    $q'_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}', q_{\text{rand}})$ 
7    $q_{\text{new}} \leftarrow \text{extend}(q'_{\text{near}}, q_{\text{rand}})$ 
8   if  $q_{\text{new}} \neq \text{null}$  and  $\text{transitionTest}(\mathcal{T}', c(q'_{\text{near}}), c(q_{\text{new}}))$  then
9      $\text{addNewNode}(\mathcal{T}', q_{\text{new}})$ 
10     $\text{addNewEdge}(\mathcal{T}', q'_{\text{near}}, q_{\text{new}})$ 
11     $(\mathcal{T}'', q''_{\text{near}}) \leftarrow \text{findNearestTree}(q_{\text{new}})$ 
12     $\mathcal{T} \leftarrow \text{attemptLink}(\mathcal{T}', q_{\text{new}}, \mathcal{T}'', q''_{\text{near}}, n)$ 
13 return  $\mathcal{T}$ 

```

When building several trees, controlling their number and the timing of connection attempts are difficult issues [104]. The number of trees can be unbounded [67, 108, 134]; it can be subjected to a pre-defined bound [57, 149, 157, 158] or implicitly limited at runtime [63, 166]. Tree roots can be sampled a priori [57] or at runtime [67, 134, 166]; they can be strategic states discovered at runtime, such as configurations in narrow passages [63, 149, 157, 158]. We focus here on the case where the number of trees is fixed and equal to the number of waypoints.

3.3.2 The Multi-T-RRT Algorithm

The pseudo-code of the Multi-T-RRT is presented in Algorithm 11. Instead of building a single tree rooted at some initial configuration, the algorithm grows n trees rooted at n given waypoints $q_{\text{init}}^k, k = 1..n$. At each iteration, a tree \mathcal{T}' is chosen for expansion in a round-robin fashion among the trees $\mathcal{T}_k, k = 1..n$. Then, an extension is attempted toward a randomly sampled configuration q_{rand} , starting from its nearest neighbor, q'_{near} , in \mathcal{T}' . We use an Extend function and not a Connect one, as recommended for the Bidirectional T-RRT in Section 3.2. If the extension succeeds and the transition test is satisfied, the new configuration q_{new} is added to \mathcal{T}' and connected to q'_{near} . Then, we look for the configuration q''_{near} (and the tree \mathcal{T}'' containing it), which is the closest to q_{new} within all trees other than \mathcal{T}' . A connection between q_{new} and q''_{near} is attempted by calling the `attemptLink` function. The exploration continues until all trees are merged into one or another stopping criterion (number of nodes, number of expansions, running time) is met.

The `attemptLink` function is the one used in the Bidirectional T-RRT (see Section 3.2). If the two configurations are closer than ten times the extension step-size, if the local path between them is collision-free, and if the configurations along this path pass the transition test, then the two trees are merged, and the number of trees is decreased by 1. This distance threshold (of ten times the extension step-size) represents a good trade-off between 1) wasting time checking edges that are unlikely to be valid (if the threshold is too high) and 2) having difficulties connecting trees (if the threshold is too low). Using the transition test of T-RRT and testing tree connections based on cost constraints enables the Multi-T-RRT to favor low-cost regions of the space, and thus to yield low-cost paths.

3.3.3 Other Multi-Tree Variants

To develop the Multi-T-RRT we addressed several points for which we had to choose among various alternatives. Note that the numerical evaluation performed to assess the performance of all these alternatives is not reported here because it is not very informative.

The first point is to decide which tree(s) to expand at a given step of the exploration process. The simplest strategy is to grow all trees at each iteration toward the same configuration q_{rand} [67, 108]. By having a single tree grown at each iteration, chosen in a round-robin fashion [57, 158], the trees are expanded toward different configurations q_{rand} , which appears to work better. Another strategy is to expand the tree that is the closest to q_{rand} [166]. However, when testing this approach, we have found that it can be difficult to expand trees that are growing close to the boundaries of the configuration space. A more sophisticated approach consists of choosing the tree to be expanded based on some probabilities that can be fixed [149] or adaptive [157]. But, we have found that such strategies show no clear benefit in terms of reducing running time or increasing path quality.

The second point is to decide when to attempt linking trees. The simplest strategy is to try after each successful expansion of a tree [34, 57, 67, 108, 158, 166]. Other, more sophisticated approaches consist of attempting a connection only when the bounding box of the expanded tree has increased in size [149], or when some stochastic test is satisfied, based on fixed or adaptive probabilities [157]. However, we have found that these approaches lead to many missed good opportunities for connection.

The third point is about how to perform the link attempt after a tree has been successfully expanded. This can involve a single tree, usually the nearest one [57, 157, 166], or a randomly chosen one. It can also involve all the other trees [34, 63, 67, 108, 149] or a subset of these trees, containing, for example, some of the closest ones and some randomly chosen ones [134]. When a tree is chosen for the link attempt, we have to decide which node in this tree we will try to connect the new node of the expanded tree to. Again, it can be the nearest one or a randomly chose one. After evaluation, we have found that random choices are not beneficial. It works better to attempt a connection between the new node and its nearest neighbor within the nearest tree.

3.3.4 Evaluation Results

We have evaluated the Multi-T-RRT on several cost-space path-planning problems that differ in terms of C-space dimensionality, geometrical complexity and cost-function type. We report results for three of them here. For each example, we define ten waypoints that have to be visited in a pre-defined order (only to facilitate the evaluation of the algorithms). The *Landscape* problem is the 2D cost-map illustrated by Fig. 3.8, in which the cost is represented by the elevation. The *Stones-large* problem (presented in Fig. 3.9) is a 2-DoF problem in which a disk has to go through a large space cluttered with rectangular-shaped stones. The objective is to maximize clearance, so the cost of a configuration is the inverse of the distance between the disk and the closest obstacle. The *Engine-multi* problem (shown in Fig. 3.10) involves a 6-DoF manipulator arm holding a sensor with a spherical extremity, used to inspect a car engine. The objective is to keep the sensor as close as possible to the engine, so the cost of a configuration is the distance between the sphere and the engine’s surface.

To fairly assess the Multi-T-RRT, no smoothing is performed on the solution paths. On all problems, we record the running time t (in seconds), the number of expansion attempts X , the number of nodes N in the produced tree, and several quality criteria evaluated on the extracted path: the average cost $avgC$, the maximal cost $maxC$, the mechanical work MW , and the integral of the cost IC (cf. Section 2.3.2). For all variables, we give values averaged over 100 runs. Note that T_{rate} is set to 0.1 for all runs. Results were obtained on an Intel Core i5 processor at 2.6 GHz with 8 GB of memory.

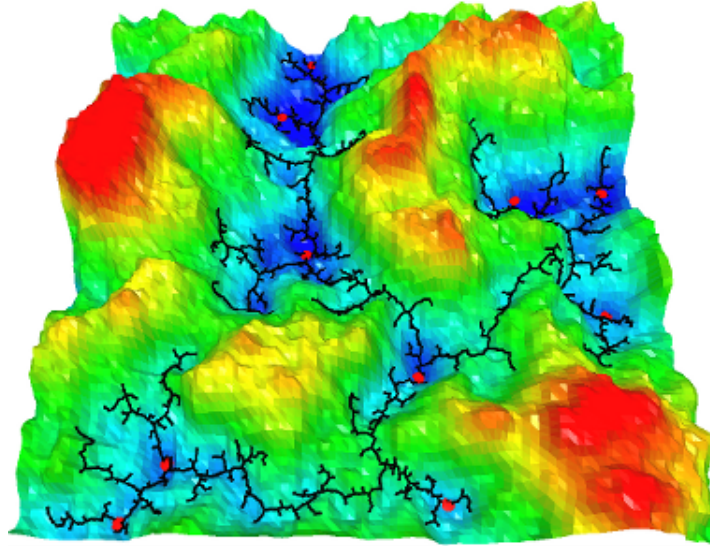


Figure 3.8: Search tree built by the Multi-T-RRT on the *Landscape* problem. On this 2D cost-map, the cost is color-coded (from blue to red) and represented by the elevation. The ten waypoints are materialized by red disks.

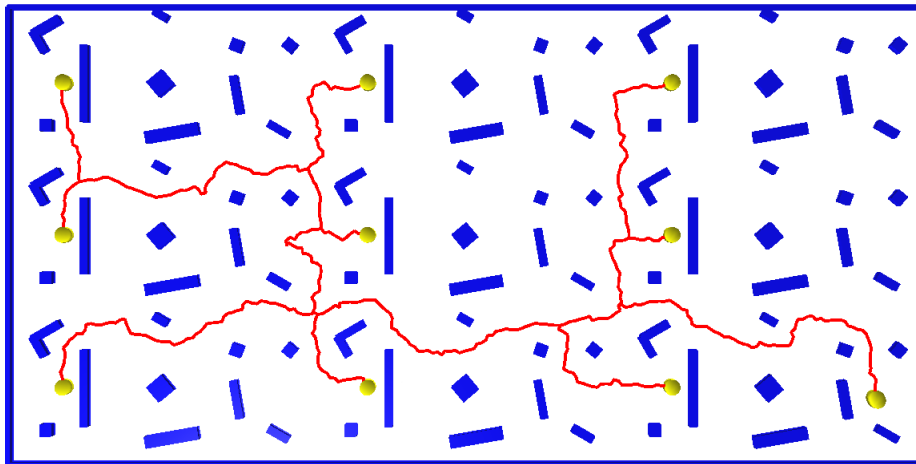


Figure 3.9: Path computed by the Multi-T-RRT on the *Stones-large* problem, and going through ten waypoints. The cost is the inverse of the distance between the 2-DoF yellow disk and the blue rectangular-shaped obstacles.

We compare two variants of the Multi-T-RRT to the basic version presented in Algorithm 11 and report the results in Table 3.3. The first variant involves having a local temperature associated to each tree, as opposed to having a global temperature. After evaluation, it seems that this modification has barely any influence on the results. The second variant is based on ensuring that all trees remain balanced (in terms of number of nodes) during the exploration. After evaluation, it is unclear whether this modification is advantageous or not. It appears to have sometimes a positive impact (e.g., on the *Stones-large* problem) and sometimes a negative impact (e.g., on the *Landscape* problem) on performance.

We also compare the Multi-T-RRT to the Bidirectional T-RRT in two ways. First, in a simple scheme involving the Bidirectional T-RRT, we compute paths between pairs of consecutive waypoints, starting from scratch each time, and concatenate them to obtain the full path visiting all waypoints. Second, in an incremental scheme involving the Bidirectional T-RRT, we compute paths between pairs of consecutive waypoints, while keeping the tree built

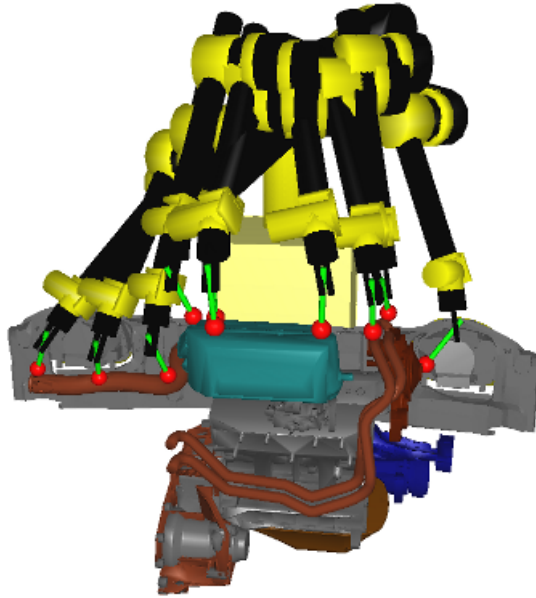


Figure 3.10: Ten waypoints defined for the *Engine-multi* problem. The 6-DoF manipulator arm holds a sensor (the red sphere) that has to follow the surface of the car engine while remaining as close to it as possible.

Table 3.3: Evaluation of the Multi-T-RRT and the Bidirectional T-RRT on three cost-space path-planning problems. The algorithm variants considered here are: 1) basic Multi-T-RRT, 2) balanced Multi-T-RRT, 3) local-temperature Multi-T-RRT, 4) simple Bidirectional T-RRT, and 5) incremental Bidirectional T-RRT.

		$avgC$	$maxC$	MW	IC	t (s)	N	X
<i>Landscape</i>	1	11	22	240	10,000	0.06	1,100	6,000
	2	11	22	230	9,900	0.18	1,300	13,000
	3	11	22	240	10,000	0.08	1,200	9,000
	4	11	22	230	9,800	0.12	2,700	20,000
	5	11	22	240	9,900	0.12	1,400	10,000
<i>Stones-large</i>	1	2.4	5.9	110	27,000	0.38	2,000	18,000
	2	2.4	5.9	110	29,000	0.28	1,100	8,000
	3	2.5	5.9	110	29,000	0.38	2,100	18,000
	4	2.3	5.9	97	25,000	0.57	4,500	38,000
	5	2.2	5.9	104	28,000	0.43	2,100	18,000
<i>Engine-multi</i>	1	4.4	19	470	20,000	0.8	500	14,000
	2	4.1	19	480	20,000	0.9	600	16,000
	3	4.3	19	470	19,000	1.0	600	18,000
	4	3.8	19	400	16,000	2.0	1,100	39,000
	5	3.7	19	540	20,000	1.5	700	28,000

so far instead of deleting it as in the simple scheme. Results obtained with these two schemes are reported in Table 3.3. As expected, the Multi-T-RRT is faster than the path planners involving the Bidirectional T-RRT. Moreover, in spite of performing a quicker exploration of the space, the Multi-T-RRT produces paths whose quality is only slightly worse than that of paths produced by the path planners involving the Bidirectional T-RRT. Therefore, the performance improvement is not achieved at the expense of path quality.

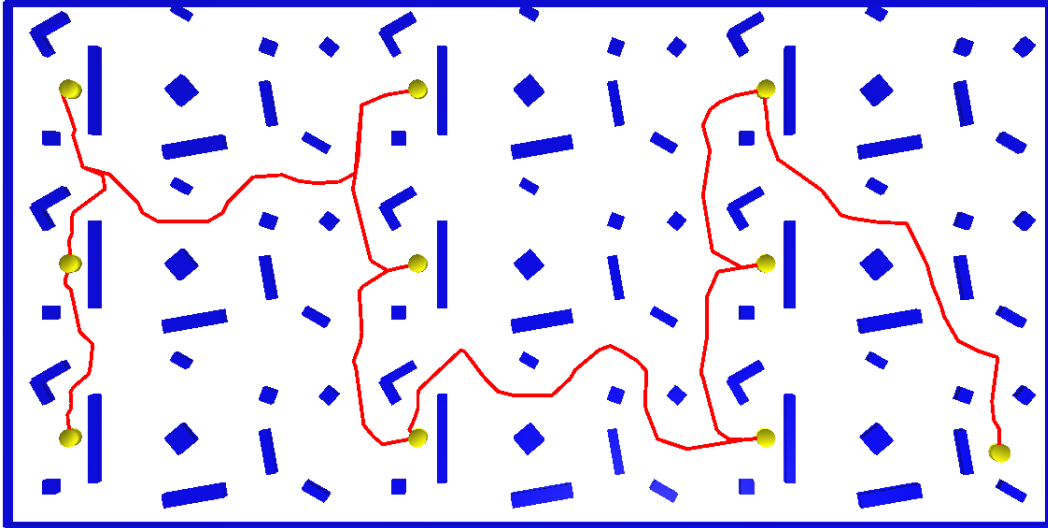


Figure 3.11: Path produced by the Multi-T-RRT enhanced with useful-cycle addition on the *Stones-large* problem, for a running time of 5 s.

3.3.5 Useful-Cycle Addition

Even though the paths that the Multi-T-RRT returns have low cost, they might not necessarily represent the most efficient way to visit a set of waypoints (e.g., see Fig. 3.9, where a trajectory would have the disk go forward and then backward along several portions of the path). To address this issue, we propose a simple approach based on the anytime paradigm and the addition of useful cycles, which allows for a continual improvement of the solution’s quality. The underlying principle is quite simple: after all trees are connected, we allow the exploration to continue and we activate a cycle-addition procedure [124]. This leads to the creation of new paths that can be of better quality than the best one found so far.

The cycle-addition procedure works as follows. When a new configuration q_{new} is added to the graph, we consider all the other configurations within a pre-specified distance of q_{new} in \mathcal{C} as potential candidate targets for new edges. Among these candidates, we are interested in those that are “close” to q_{new} in \mathcal{C} , but “far” from q_{new} in the graph. For each candidate q_c , if the cost of the local path between q_{new} and q_c in \mathcal{C} is strictly less than the cost of the lowest-cost path between q_{new} and q_c in the graph, we add an edge between q_c and q_{new} , thus creating a useful cycle. More details on this cycle-addition procedure and on the anytime variant of T-RRT are presented in Chapter 4.

As an example, Fig. 3.11 shows the benefits of using this Multi-T-RRT with useful-cycle addition on the *Stones-large* problem. The improvement is especially visible when this figure is compared to Fig. 3.9. The path in Fig. 3.11 appears smoother and does not feature so many portions along which the disk would go forward and then backward. Note that this path is representative of what we obtain for a running time of 5 s (as observed over 100 runs). Its global (IC) cost is about half the cost of the path in Fig. 3.9 (obtained in 0.3 s).

To show what the Multi-T-RRT can achieve in realistic settings, we now introduce an additional cost-space path-planning problem. It is an industrial inspection problem involving an aerial robot in a dense environment, as illustrated by Fig. 3.12. In this *Inspection* example, a quadrotor is used to inspect an oil platform, by going through eight waypoints whose order is defined a priori. The context of this example is the same as what is presented in Section 3.2.3. In order to maximize clearance, the cost of a configuration is defined as the inverse of the distance between the quadrotor and the obstacles. Even though this example features a large-scale workspace, the Multi-T-RRT can quickly provide a solution path: in about 5 s on average over 100 runs.

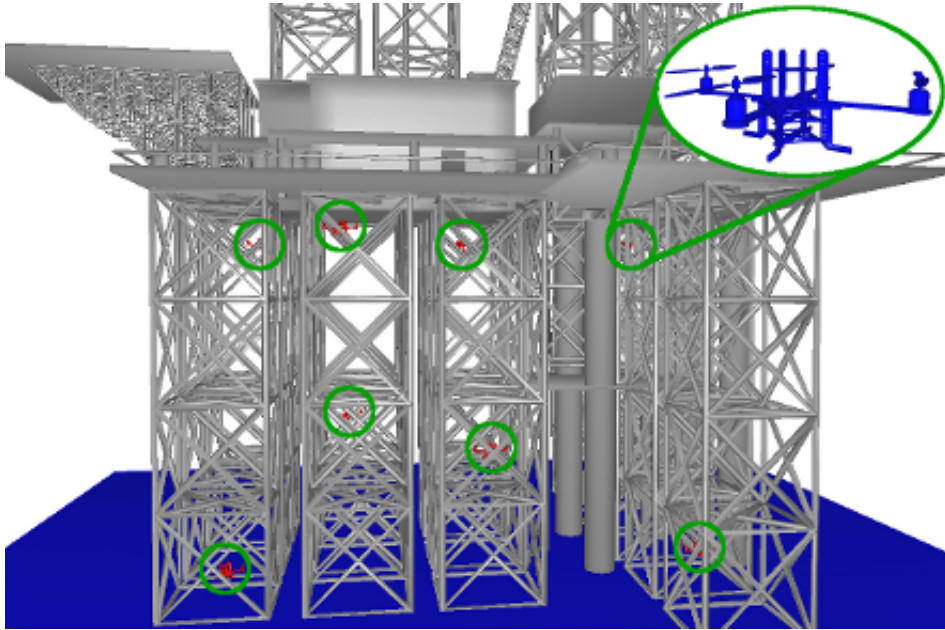


Figure 3.12: Eight waypoints (shown in red and circled in green) defined for a quadrotor (whose close-up is shown in blue) inspecting an oil platform.

Finally, we perform an experiment aimed at quantifying the efficiency of the useful-cycle addition procedure. For that, we evaluate the evolution over time of the quality of the solution paths produced by the Multi-T-RRT on the four cost-space path-planning problems studied in this section (see Fig. 3.13). On the *Landscape* problem, path quality is measured by the mechanical work, MW , and on the other problems it is measured by the integral of the cost, IC . To clarify this choice, on each problem we have picked the criterion producing the most “natural-looking” optimal path: MW is interesting on 2D cost-maps because it avoids going through high-cost saddles; IC is interesting in realistic problems because it produces shorter paths (that intuitively look better). Note that we use two different quality criteria in order not to bias our results by working only with a single one. We compare the rate of convergence of the quality of paths produced by the Multi-T-RRT with useful-cycle addition to the rates of convergence observed when planning with versions of PRM [124] and $RRT_{\text{obst way}}$ [57] creating cycles. Fig. 3.13 shows that the Multi-T-RRT yields a slightly better rate of convergence than $RRT_{\text{obst way}}$. The poor results of PRM, especially in high-dimensional spaces, are due to the fact that it does not involve any cost constraint. As it features a similar convergence rate, IRS [117] would probably not perform better. RRT^* [93] would provide a better point of comparison, but it would require a multi-tree extension.

3.4 Conclusion

In this chapter, we have presented several enhancements to the T-RRT algorithm. We have evaluated them on various cost-space path-planning problems that differ in terms of geometrical complexity, configuration-space dimensionality and cost-function type.

First, we have focused on the mono-directional variant of T-RRT. We have explained how its transition test can be modified to improve performance, mainly by optimizing the adaptive behavior of the temperature parameter. We have also studied two extensions of T-RRT inspired by classical RRT variants: the Goal-biased T-RRT and the Connect T-RRT. We have shown that both are delicate to use because, despite reducing running time, they sometimes decrease path quality.

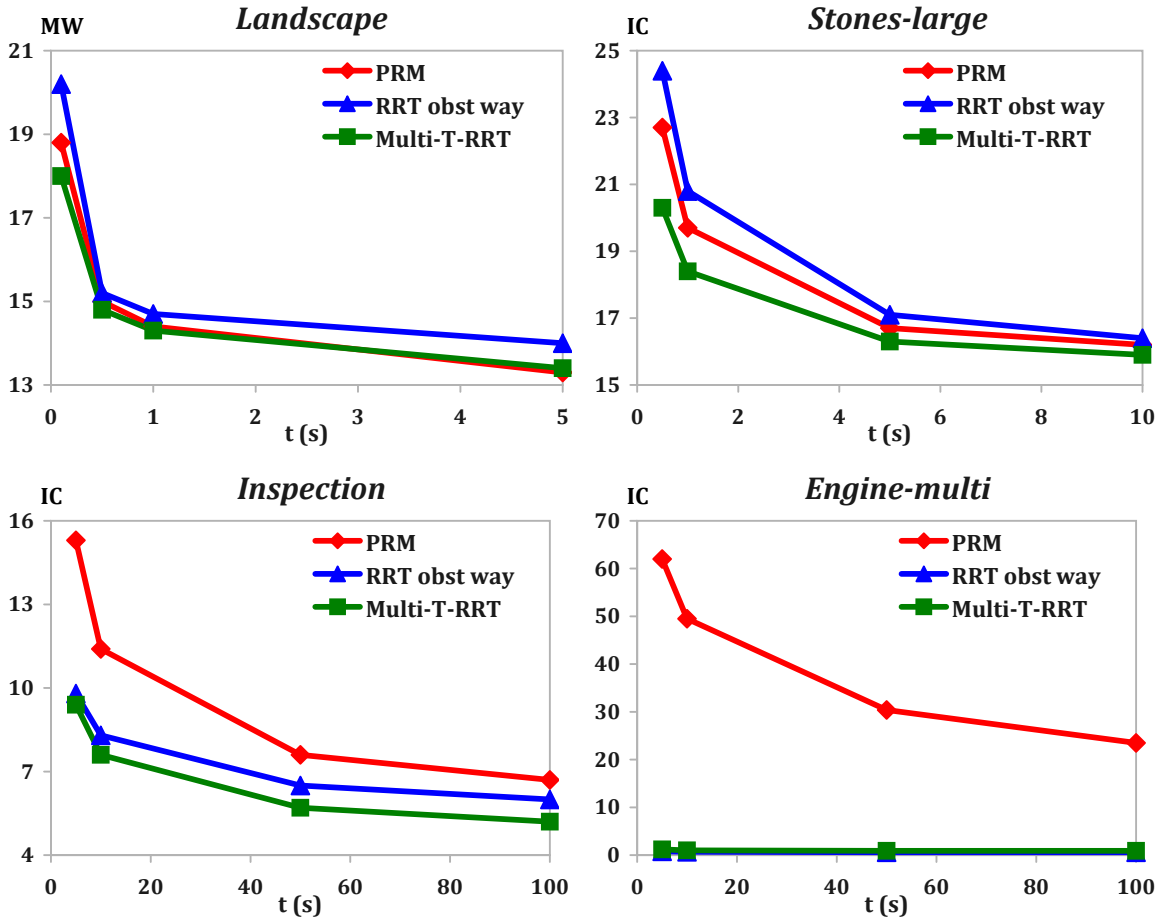


Figure 3.13: Evolution of the quality (measured by IC or MW) of paths produced by the Multi-T-RRT, $RRT_{\text{obst way}}$ and PRM over time, on four cost-space path-planning problems of increasing dimensionality.

Second, we have proposed a Bidirectional variant of T-RRT. Contrary to the development of the Bidirectional RRT, which was rather straightforward, the development of an efficient Bidirectional T-RRT is more challenging, because of the subtleties resulting from dealing with cost constraints. We have shown that our Bidirectional T-RRT improves performance when compared to the Extend T-RRT, mainly in terms of success rate and running time, but also often in terms of path quality, and we have presented some clues as to why this is so. The Bidirectional T-RRT does not always outperform the Goal-biased and Connect T-RRT, but it provides more consistent results and is therefore a better choice. We have also illustrated the need to enhance T-RRT this way with a realistic industrial inspection problem involving an aerial robot. In such context, the Extend T-RRT cannot find a solution in a reasonable amount of time, contrary to the Bidirectional T-RRT.

Third, we have presented a multi-tree variant of T-RRT named *Multi-T-RRT*. To achieve the highest efficiency, we have selected the best techniques among those involved in other multi-tree sampling-based path planners. When looking for a path going through a given set of waypoints, we have shown that the Multi-T-RRT is faster than path planners based on the Bidirectional T-RRT, and that it yields paths of similar quality. We have also given preliminary explanations on how to devise an anytime variant of the Multi-T-RRT by enhancing it with a useful-cycle addition procedure that enables the solution to be continually improved. Finally, we have shown that this anytime variant of the Multi-T-RRT yields a better convergence rate for solution-path quality than PRM and $RRT_{\text{obst way}}$.

Developing an anytime variant of T-RRT to solve the optimal path planning problem is part of what is presented in Chapter 4. In that chapter, we provide more details on the useful-cycle addition procedure involved in this *Anytime* T-RRT. We also prove that this algorithm is asymptotically optimal, meaning that it offers theoretical guarantees of convergence toward the optimal solution-path. Then, later in this thesis, we integrate the Anytime T-RRT and the Multi-T-RRT introduced in this chapter to develop an *Anytime Multi-T-RRT*. This algorithm is applied to complex robotics problems that require to find the most efficient way to visit a set of waypoints (cf. Chapter 6). It is also applied to the structural biology issue of exploring the energy landscape of peptides (cf. Chapter 7).

As future work, we could investigate further improvements and extensions of T-RRT. In particular, more sophisticated heuristics for the adaptive variation of the temperature parameter could lead to a faster exploration while maintaining path quality. On another front, the approach we have proposed to compute a path going through a set of waypoints is a general one. It would be interesting to enhance other sampling-based path planners in a similar way and compare their performance to that of the Multi-T-RRT. The poor results obtained with PRM highlight that such path planners should involve cost constraints. Therefore, a good candidate seems to be RRT* [93], in a multiple-tree version.

Chapter 4

Efficient Optimal Path Planning in Complex Continuous Cost Spaces

In robotics, computing an optimal solution-path to a path planning problem (with respect to a given path-quality criterion) has traditionally been done by using grid-based methods such as A* or D* [148]. However, it is well-known that these methods are limited to problems involving low-dimensional search spaces. In the context of sampling-based path planning, looking for an optimal solution-path is a relatively new topic of research [93]. The first sampling-based path planners to be developed for that purpose were a variant of PRM called PRM* and two variants of RRT called RRG and RRT*. In fact, PRM*, RRG and RRT* have been proven to guarantee asymptotic optimality and, therefore, to solve the optimal path-planning problem (cf. Section 2.3). On the other hand, it has been shown that RRT (and by extension, T-RRT) offers no asymptotic optimality guarantee [93].

Historically, the first quality criteria to be optimized were path length and path duration. In this thesis, we aim at dealing with path-quality criteria defined on the basis of continuous configuration-cost functions, which is more challenging. In this context, 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 (cf. Section 3.2 and [79]). On the other hand, thanks to the filtering properties of its Metropolis-like transition test, the exploration performed by T-RRT favors low-cost regions of the configuration space. In fact, T-RRT mostly creates new nodes in these favorable areas. Nevertheless, T-RRT does not take the 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. To summarize, we can see that RRT* and T-RRT build on different (and even complementary) concepts that it is interesting to combine.

In this chapter, 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 aforementioned extensions of RRT, namely T-RRT and RRT*. 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, 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, meaning that they can solve the optimal path planning problem. They allow us to efficiently explore complex continuous cost-spaces, yielding high-quality solution paths that improve with time in an anytime fashion.

We evaluate T-RRT* and AT-RRT on several optimal path-planning problems, and show that they converge toward the optimal path faster than RRT* does. Thanks to the filtering properties of the transition test, T-RRT* and AT-RRT can efficiently solve difficult problems on which RRT* converges very slowly. We present several examples illustrating various factors that make an optimal path-planning problem difficult to solve. 1) If the problem features a large-scale workspace, even in low dimension, favoring low-cost areas 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 sub-optimal 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.

4.1 Algorithms

In this section, we combine the beneficial concepts underlying RRT* and T-RRT to devise novel algorithms inheriting their respective strengths and addressing their respective limitations. The first algorithm, called *Transition-based RRT** (T-RRT*), consists of integrating the Metropolis-like transition test of T-RRT into RRT*. The motivation is to favor the exploration of low-cost regions of the space by taking the configuration-cost function into account (as T-RRT does), while maintaining the asymptotic properties 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, as is done for PRM in [124]. The motivation is to quickly obtain a first high-quality solution-path and, then, continue 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 [60]. The sampling process of this Anytime RRT was later improved [164]. Another anytime variant of RRT builds on the idea of pruning low-quality branches during the exploration [1]. 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 [93]. RRT* enhances RRT with an anytime behavior by rewiring (i.e. removing and creating connections in) the tree built by RRT [93]. 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 [5]. The meta-approach proposed in [112], based on short-cutting and path hybridization, can also provide RRT-like algorithms with an anytime behavior. 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.

4.1.1 Transition-based RRT* (T-RRT*)

The pseudo-code of T-RRT* is presented in Algorithm 12. It extends RRT* by integrating the transition test of T-RRT (line 6). 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} . 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. Finally, since

Algorithm 12: Transition-based RRT* (T-RRT*)

input : the optimal path planning problem $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}}, c, c_p)$
the dimension d of the \mathcal{C} -space
the γ constant derived from the volume of $\mathcal{C}_{\text{free}}$

output: the graph \mathcal{G}

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

Algorithm 13: 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** $c_j \leq c_i$ **then**
- 2 **return** True
- 3 **if** $\exp(-(c_j - c_i) / T) > 0.5$ **then**
- 4 $T \leftarrow T / 2^{(c_j - c_i) / (0.1 \cdot \text{costRange}(\mathcal{G}))}$
- 5 **return** True
- 6 **else**
- 7 $T \leftarrow T \cdot 2^{T_{\text{rate}}}$
- 8 **return** False

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 $q_n \in Q_{\text{near}}$, 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} .

Compared to the version of the transition test presented in Section 3.1.2, the one used here (and shown in Algorithm 13) features a minor modification: it does not involve the cost threshold c_{max} . This ensures that no configuration sampled in \mathcal{C} has a probability zero to be accepted. Therefore, the samples produced by T-RRT* can be considered to be drawn from a continuous probability distribution with density bounded away from zero. This property is important to ensure that T-RRT* is asymptotically optimal (see Section 4.2).

Algorithm 14: Anytime Transition-based RRT (AT-RRT)

```

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

```

Algorithm 15: addUsefulCycles ($\mathcal{G}, q_{\text{new}}, c_p$)

```

input: the dimension  $d$  of the  $\mathcal{C}$ -space
        the  $\gamma$  constant derived from the volume of  $\mathcal{C}_{\text{free}}$ 
1  $n \leftarrow \text{numberOfNodes}(\mathcal{G})$ 
2  $Q_{\text{near}} \leftarrow \text{findNodesInBall}(\mathcal{G}, q_{\text{new}}, \gamma (\log(n) / n)^{1/d})$ 
3 foreach  $q_n \in Q_{\text{near}}$  do
4    $\pi_g \leftarrow \text{pathInGraph}(\mathcal{G}, q_{\text{new}}, q_n)$ 
5    $\pi_s \leftarrow \text{pathInSpace}(q_{\text{new}}, q_n)$ 
6   if  $c_p(\pi_s) < c_p(\pi_g)$  and isCollisionFree( $\pi_s$ ) then
7     addNewEdge( $\mathcal{G}, q_{\text{new}}, q_n$ )

```

4.1.2 Anytime Transition-based RRT (AT-RRT)

The pseudo-code of AT-RRT is shown in Algorithm 14. It also features the `transitionTest` (line 6) shown in Algorithm 13, and extends T-RRT by offering an anytime behavior. Before any solution path 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 [68, 124]. 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 15. 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 $\mathcal{C}_{\text{free}}$, as is done for RRT*. Furthermore, 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 $q_n \in Q_{\text{near}}$, 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 if π_s is collision-free, we add an edge to \mathcal{G} between q_{new} and q_n , thus creating a useful cycle.

4.2 Theoretical Analysis

We now review the properties of the T-RRT* and AT-RRT algorithms, in terms of probabilistic completeness (cf. Section 2.1) and asymptotic optimality (cf. Section 2.3).

It has already been proven in previous work that the T-RRT and RRT* algorithms are probabilistically complete [83,93]. 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 6 (Probabilistic completeness of T-RRT*). *The T-RRT* algorithm is probabilistically complete.*

Theorem 7 (Probabilistic completeness of AT-RRT). *The AT-RRT algorithm is probabilistically complete.*

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

$$\gamma > 2 \left(1 + \frac{1}{d}\right)^{\frac{1}{d}} \left(\frac{\mu(\mathcal{C}_{\text{free}})}{\zeta_d}\right)^{\frac{1}{d}}, \quad (4.1)$$

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 $\mu(\cdot)$ is an operator measuring the volume of a space. Under this assumption, it has been proven that RRT* is asymptotically optimal [93].

The only difference between T-RRT* and RRT* is the addition of the transition test, that filters the configurations added to \mathcal{G} based on their costs. The consequence of applying such rejection sampling is that the samples cannot be assumed to be drawn from a uniform distribution on \mathcal{C} . However, as already mentioned, they can be assumed to be drawn from a continuous probability distribution with density bounded away from zero. Even though the asymptotic optimality of RRT* was proven under a “uniform distribution” assumption, this result can be extended to any continuous probability distribution with density bounded away from zero [93]. Therefore, T-RRT* is also asymptotically optimal.

Theorem 8 (Asymptotic optimality of T-RRT*). *The T-RRT* algorithm is asymptotically optimal.*

It is clear that 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 the two algorithms lies in the management of edges. First, contrary to T-RRT*, in AT-RRT no edge is removed, thus leading to the creation of cycles, but this has no impact on the current analysis. The main point is that both algorithms make the decisions to create alternative paths based on cost improvement. Where they differ is on the criterion that an edge has to satisfy to be considered useful in terms of cost improvement. In T-RRT*, this criterion is based on whether an edge allows a configuration to be connected to q_{init} via a path in \mathcal{G} having minimal cost. In AT-RRT, this criterion is based on whether an edge allows two configurations to be connected via a path in \mathcal{C} whose cost is lower than the costs of the existing paths in \mathcal{G} . It is clear that both criteria achieve the same goal: allowing the cost of the solution path to decrease with time. Finally, since the cost-based decisions on the addition of useful cycles happen in neighborhoods of radii based on a value of γ satisfying (4.1), AT-RRT is also asymptotically optimal.

Theorem 9 (Asymptotic optimality of AT-RRT). *The AT-RRT algorithm is asymptotically optimal.*

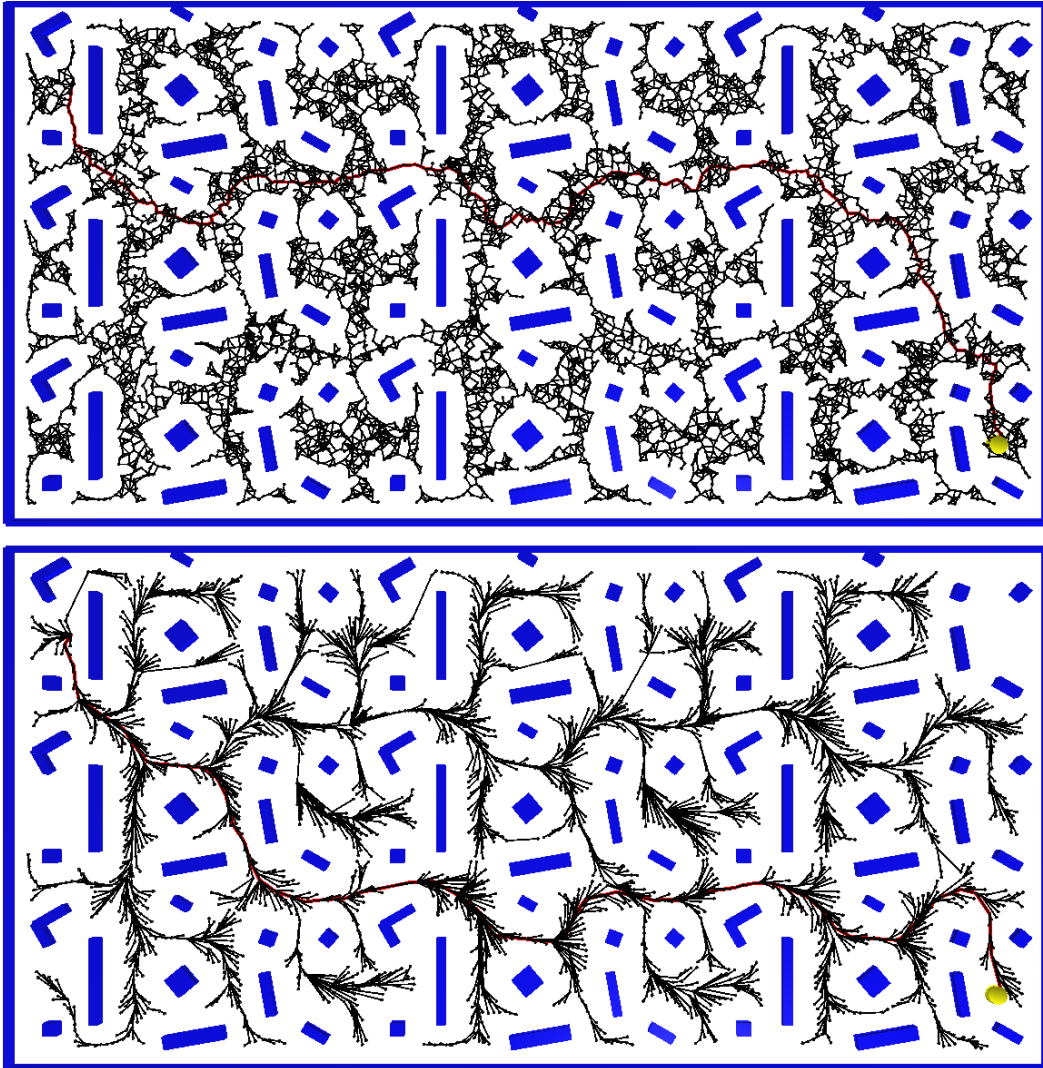


Figure 4.1: *Stones-large* problem: 2-DoF yellow disk moving among blue rectangular-shaped obstacles, while having to maximize its clearance. The figure shows graphs built by AT-RRT (top) and T-RRT* (bottom) after a running time of 0.5 s.

4.3 Evaluation

4.3.1 Path Planning Problems

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

The *Stones-large* problem (illustrated in Fig. 4.1, Fig. 4.5 and Fig. 4.9) is a 2-DoF 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.

The *Inspection* problem deals with industrial inspection in a dense environment, and involves an aerial robot, as shown in Fig. 4.2 and Fig. 4.6. 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 controllability 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.

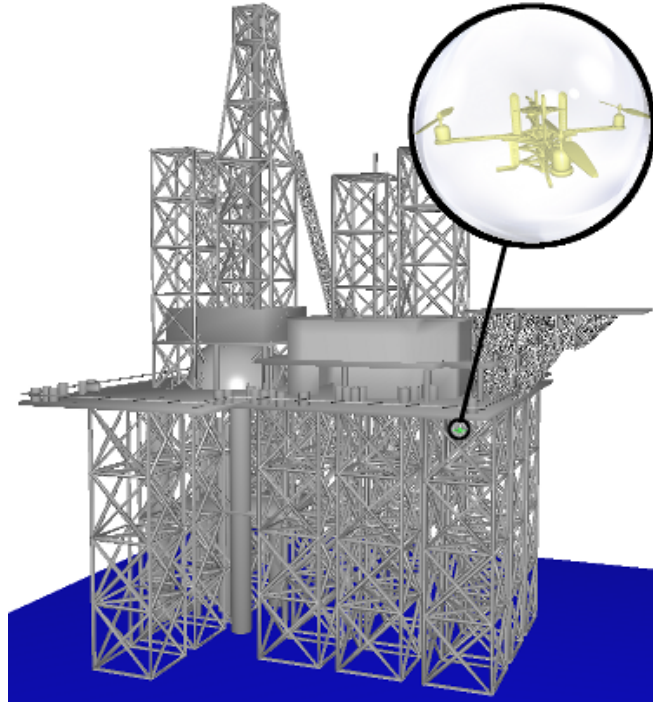


Figure 4.2: *Inspection* problem: quadrotor (whose close-up is shown in yellow) inspecting an oil platform. The cost function involves the clearance of the quadrotor’s 3-DoF safety sphere.

The *Transport* problem features aerial robots, and deals with the collaborative transport of objects, as shown in Fig. 4.3. 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 passive 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 will 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 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 going through the lower one requires the robotic system to tilt sharply.

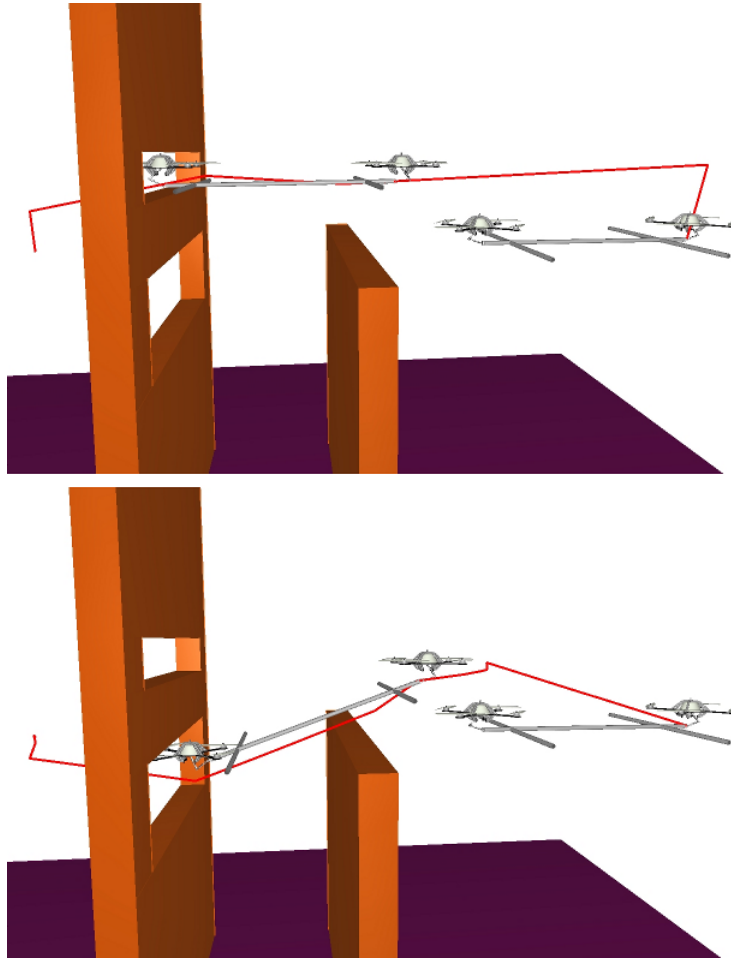


Figure 4.3: *Transport* problem: the two quadrotors have to transport an object and go through one of the holes in the wall, while maintaining the balance of the whole system. Both images show an intermediate configuration as well as the goal configuration along paths obtained after 50 s. Top: path produced 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.

The *Snake* problem (illustrated in Fig. 4.4) 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 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 consecutive revolute joints, added to the absolute value of the first revolute joint. The objective is to favor a straight configuration of the robot, or configurations in which all revolute joints have the same value, which correspond to a regular coiling of the robot. This problem will enable us to analyze the behavior of the algorithms in higher dimension.

4.3.2 Settings

Before applying T-RRT* and AT-RRT to these path planning problems, the values of their parameters have to be set. First, T_{rate} is set to 0.1 and T is initialized to 10^{-6} . Finding a good value for γ happens to be a real issue. As already mentioned, the lower bound for γ expressed in (4.1) 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

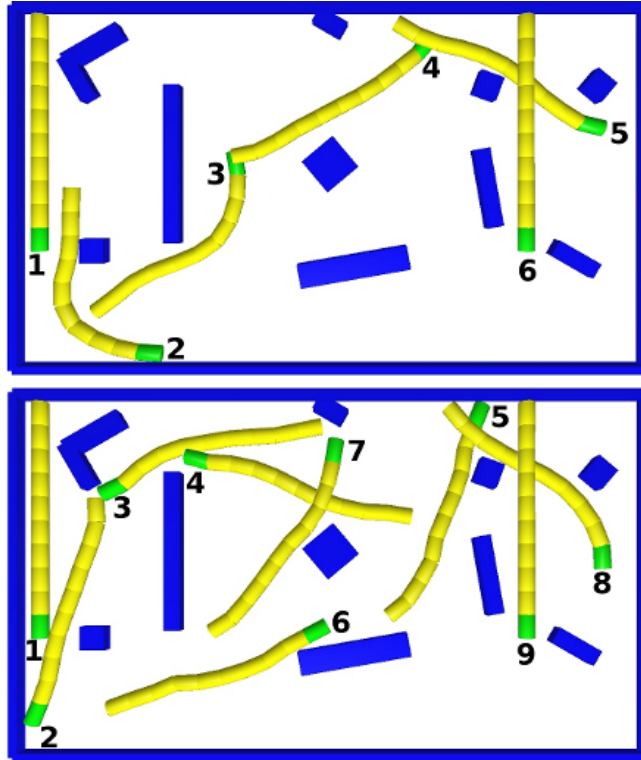


Figure 4.4: Selected configurations along paths produced by AT-RRT when minimizing IC (top) or MW (bottom), after a running time of 100 s, on the *Snake* problem. A snake-like object has to move among obstacles while remaining as straight as possible. The cost function favors regular over irregular coiling of the object. T-RRT* provides similar results.

with simple geometric models, but this is not realistic otherwise. To ensure that γ satisfies (4.1), we use the value

$$\gamma = 2 \left(1 + \frac{1}{d} \right)^{\frac{1}{d}} \left(\frac{\mu(\mathcal{C})}{\zeta_d} \right)^{\frac{1}{d}}. \quad (4.2)$$

On the *Stones-large* and *Inspection* problems, since \mathcal{C} is an Euclidean space, its volume can easily be computed using the validity interval of every DoF. However, this is not straightforward on the *Transport* and *Snake* problems because of the revolute joints. 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 motion planning platform *Move3D*. To fairly assess them, no smoothing is performed on the solution paths. In this evaluation, to assess path quality, we use both IC and MW not to limit ourselves to a single criterion, which could bias the interpretation of the results. However, comparing cost criteria is out of the scope of this evaluation. Note that values of IC and MW are averaged over 100 runs. Results have been obtained on an Intel Core i5 processor at 2.6 GHz with 8 GB of memory.

4.3.3 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. 4.1 on the *Stones* problem. The upper figure clearly shows the cycles created by AT-RRT, and the redundancy in paths. As can be 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

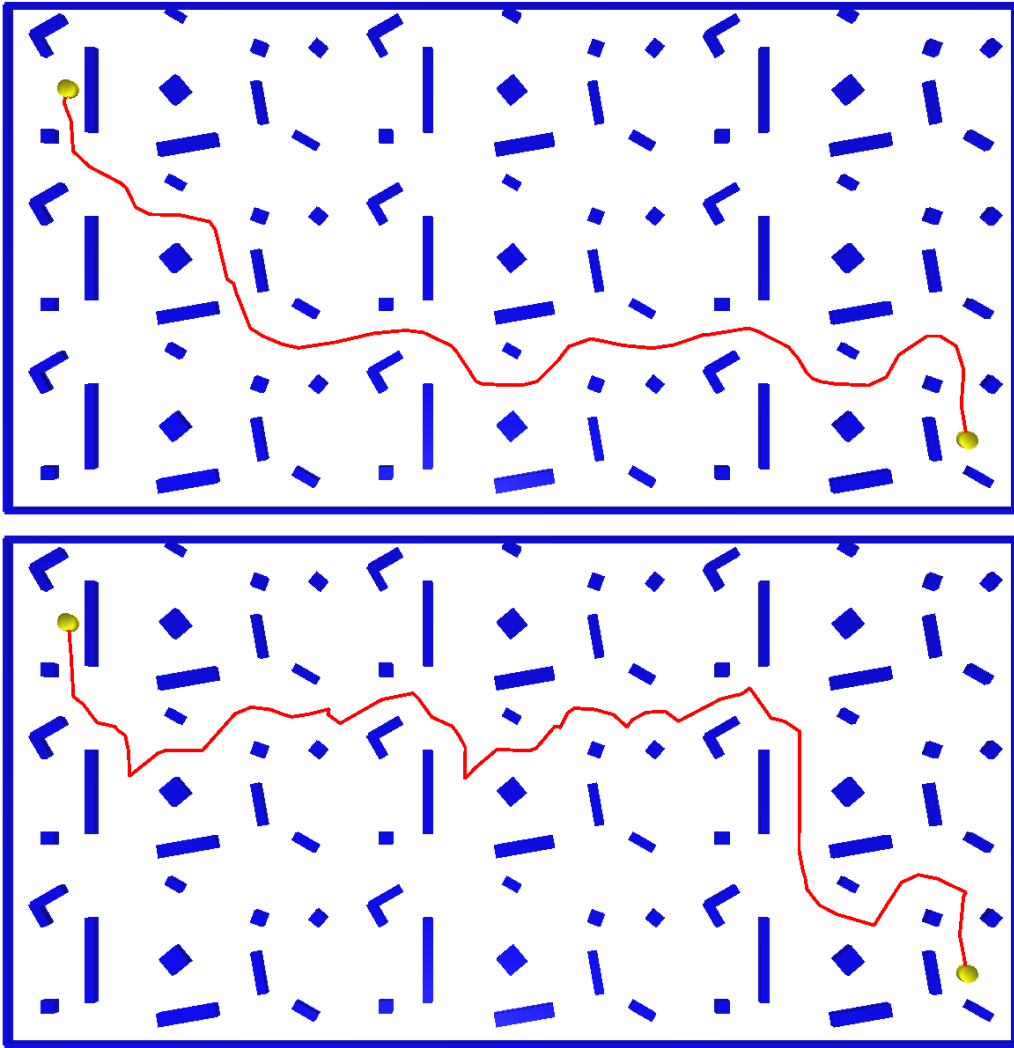


Figure 4.5: Solution paths produced by T-RRT* on the *Stones-large* problem when minimizing IC (top) or MW (bottom) after a running time of 10 s. Solution paths produced by AT-RRT are similar.

these differences in behavior do not create significant 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 cost criterion: IC or MW. This is clearly visible in Fig. 4.5 and 4.6. Minimizing IC tends to favor shorter paths along which the maximal cost can be very high (as illustrated by the cost profiles in the lower part of Fig. 4.6), and minimizing MW sometimes produces strangely convoluted paths (as illustrated by the lower part of Fig. 4.5). 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 thesis. 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*. 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* [87]. The *branch-and-bound* heuristic consists of trimming the

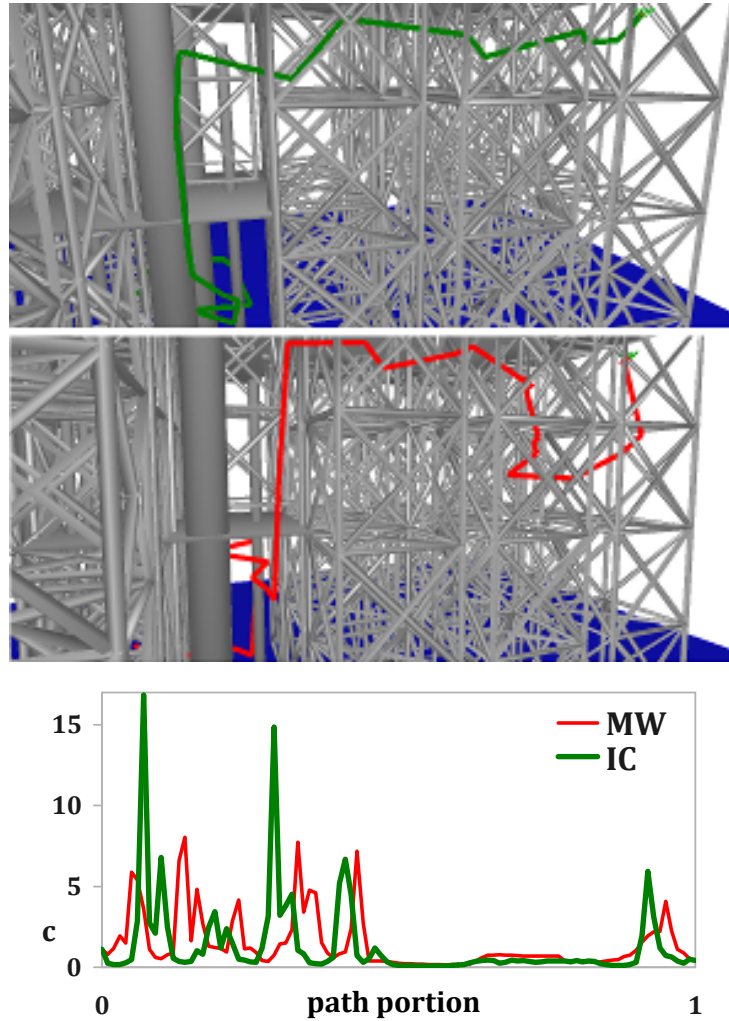


Figure 4.6: Paths produced by AT-RRT on the *Inspection* problem when minimizing IC (top) or MW (middle), after a running time of 10 s. The cost profiles of the two paths are also shown (bottom). Paths produced by T-RRT* are similar.

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 [94]. 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 $(q_{\text{init}}, q_{\text{goal}})$ of configurations) are reported in Fig. 4.7 for IC, and Fig. 4.8 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-large*, 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 with RRT, cf. Fig. 4.9), contrary to graphs produced by T-RRT* or AT-RRT (cf. Fig. 4.1).

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

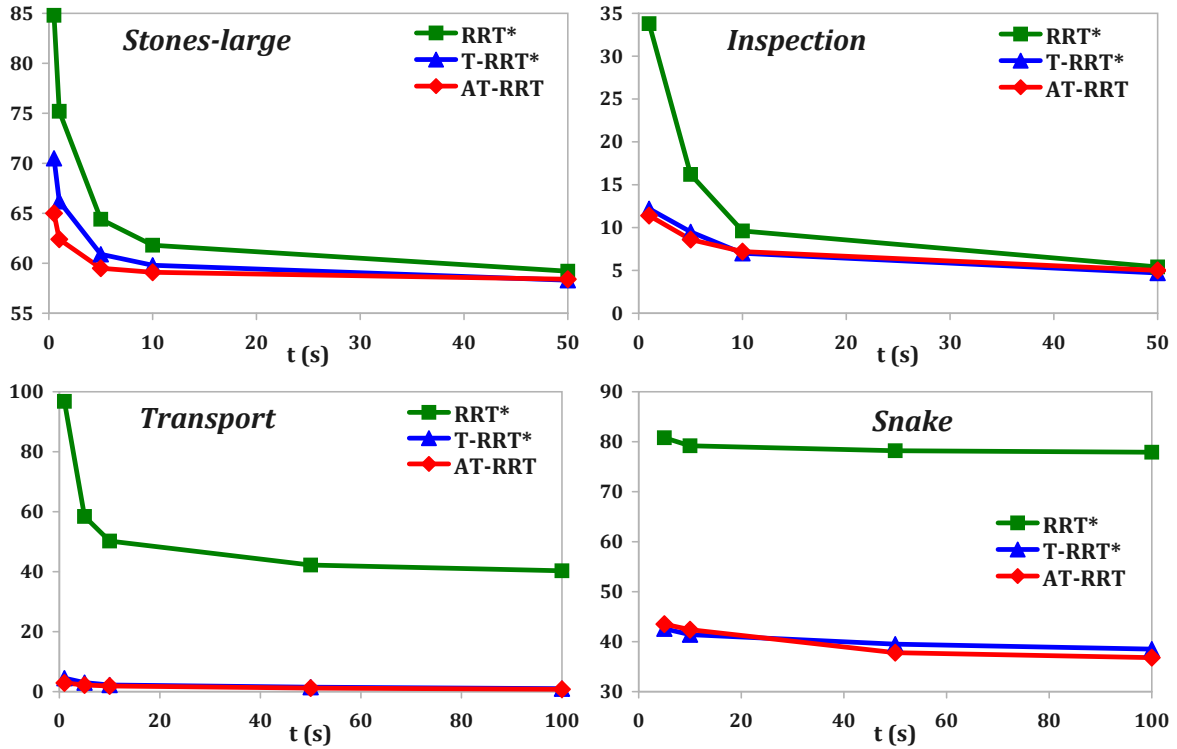


Figure 4.7: Evolution over time of the costs (IC) of the solution paths produced by RRT*, T-RRT* and AT-RRT, on the four path planning problems.

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. 4.3). On the contrary, the first solution produced by RRT* can belong to any of the two homotopic classes; if it is found in the sub-optimal one (i.e. the lower hole), RRT* gets stuck in this class and into optimizing a low-quality solution (as shown in the lower part of Fig. 4.3).

On high-dimensional problems, such as *Snake*, RRT* usually converges very slowly. Looking at Fig. 4.7 and Fig. 4.8, one may think that this is also the case for T-RRT* and AT-RRT. To check that, we have let all 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. 4.8, 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 because they are not very informative) are similar to what we report above.

4.4 Conclusion

In this chapter, we have proposed two new sampling-based algorithms to solve the optimal path planning problem by combining the underlying principles of T-RRT and RRT*, the goal being to benefit from their respective strengths while overcoming 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

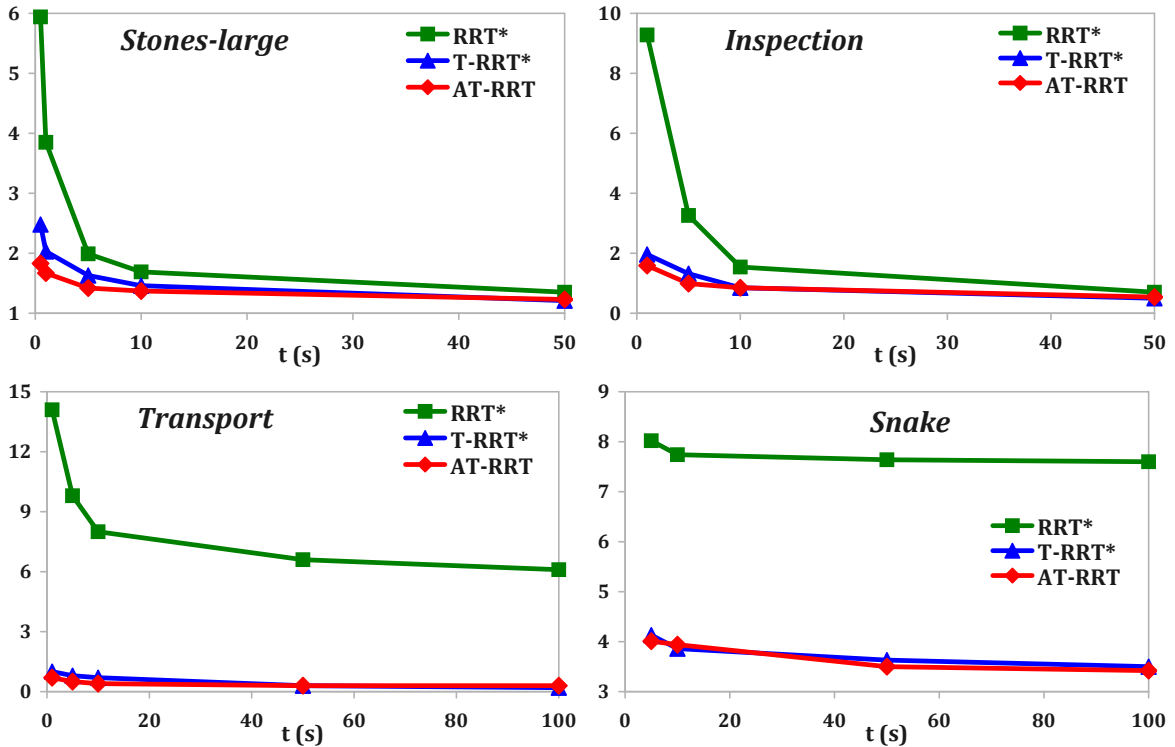


Figure 4.8: Evolution over time of the costs (MW) of the solution paths produced by RRT*, T-RRT* and AT-RRT, on the four path planning problems.

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 tend to show that AT-RRT performs 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 builds a graph containing cycles, therefore providing alternative paths over the configuration space. This could be leveraged when path replanning is required due to errors in the model or moving obstacles.

Another interesting aspect of AT-RRT is that it can easily be extended into a multiple-tree algorithm. This is what we do later in this thesis, based on the multiple-tree variant of T-RRT presented in Chapter 3. We develop an *Anytime Multi-T-RRT* that we apply to difficult problems in robotics and structural biology. On the robotics side, we apply this algorithm to “ordering-and-pathfinding” problems (cf. Chapter 6). On the structural biology side, we apply it to the exploration of energy landscapes of peptides (cf. Chapter 7).

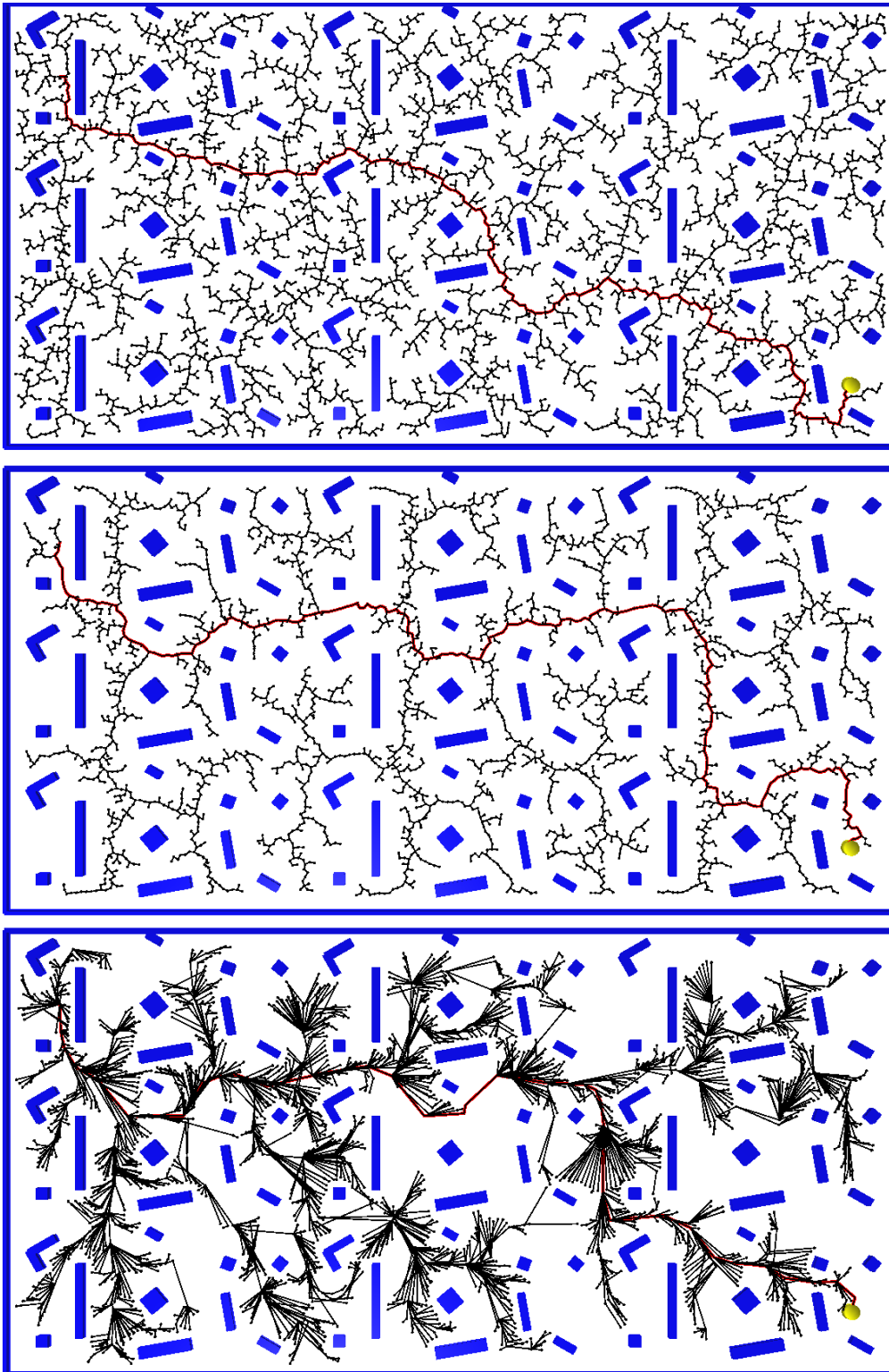


Figure 4.9: Graphs built on the *Stones-large* problem, by RRT (top), T-RRT (middle) and RRT* (bottom). As this example involves a clearance-based cost function, nodes close to the obstacles have high costs, and nodes far from the obstacles have low costs. Because they do not take this configuration-cost 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. 4.1).

Chapter 5

Parallel Path Planning on Distributed-Memory Architectures

Despite the successes achieved in sampling-based path planning, some difficult problems still remain too challenging for current methods. One way to solve this issue is to propose enhanced versions of existing approaches, at an algorithmic level, as we have done in Chapters 3 and 4. Another solution is to optimize existing approaches, at an implementation level, to make them more computationally efficient. In the context of RRT-like algorithms, several techniques have been proposed, such as reducing the complexity of the nearest neighbor search [162], dynamically controlling sampling domains [84], or reducing the dispersion of the samples [109]. Instead of optimizing the different components of these algorithms, a more global approach can be adopted by building on parallel computing.

In this chapter, we address the issue of improving the performance of RRT-like algorithms by exploiting the speedup inherent to parallel computation. Some results have been obtained in that direction (cf. Section 2.4). However, existing work considers mainly shared-memory architectures and small-scale parallelism, up to 16 processors. In this work, we are interested in what can be achieved by larger-scale parallelism. We focus on parallelizing RRT-like algorithms on large-scale distributed-memory architectures, which requires using the classical Message Passing Interface (MPI).

This work on parallelization can be beneficial to feasible, cost-space and optimal path planning. Nevertheless, the examples we present in this chapter focus on feasible and cost-space path planning. For sake of clarity, we start by proposing parallel versions of the basic mono-directional variant of RRT. Then, we perform experimental evaluations involving both the RRT and T-RRT algorithms. Finally, we discuss how the novel RRT-like algorithms introduced in other chapters, such as the (anytime) multiple-tree variant of T-RRT, can be parallelized (cf. Section 5.4).

In what follows, we compare three parallel versions of RRT that are based on well-known parallelization schemes: *OR parallel* RRT, *Distributed* RRT and *Manager-worker* RRT. In addition to the algorithms themselves, we present the main technicalities involved in their development (Section 5.1). Our contribution focuses on evaluating these algorithms on several path planning problems and showing their differences in behavior (Section 5.2). We also analyze their performance in order to understand the impact of several characteristics of the studied problems (Section 5.3). Our evaluations show that parallelizing RRT-like algorithms with MPI can provide substantial performance improvement in two cases: 1) problems for which the variability in sequential runtime is high can benefit from the OR parallel RRT; 2) problems for which the computational cost of an RRT expansion is high can benefit from the Distributed RRT and Manager-worker RRT. Virtually any problem in structural biology and many problems in robotics are characterized by computationally-expensive RRT expansions and can thus benefit from these parallel algorithms.

Algorithm 16: OR parallel RRT

```

input : the feasible path planning problem  $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$ 
output: the tree  $\mathcal{T}$ 
1  $\mathcal{T} \leftarrow \text{initTree}(q_{\text{init}})$ 
2 while not (stoppingCriteria $(\mathcal{T})$  or received $(\text{endMsg})$ ) do
3    $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
4    $q_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}, q_{\text{rand}})$ 
5    $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}, q_{\text{rand}})$ 
6   if  $q_{\text{new}} \neq \text{null}$  then
7      $\text{addNewNode}(\mathcal{T}, q_{\text{new}})$ 
8      $\text{addNewEdge}(\mathcal{T}, q_{\text{near}}, q_{\text{new}})$ 
9 if stoppingCriteria $(\mathcal{T})$  then
10   $\text{broadcast}(\text{endMsg})$ 
11  return  $\mathcal{T}$ 

```

5.1 Parallelization of RRT

Many algorithms based on RRT or similar to RRT can be parallelized using the schemes we propose here or slight adaptations of them. For clarity’s sake, we present these schemes as parallel versions of the mono-directional RRT in its Extend version. Note that some of the variants of RRT used in this thesis, such as ML-RRT or T-RRT, can be parallelized using the proposed schemes as such.

For scalability purposes, we parallelize RRT on distributed-memory architectures, using the message passing paradigm [85], one of the most widespread approaches in parallel programming. Since this paradigm imposes no requirement on the underlying hardware and requires to explicitly parallelize algorithms, it enables a wide portability: any algorithm developed following this approach can also run on shared memory. Besides, scalable distributed-memory architectures are rather commonly available, in the form of networks of personal computers, clustered workstations or grid computers. To develop our parallel algorithms, we have chosen to comply with the standard and widely-used Message Passing Interface¹ (MPI). Its logical view of the hardware architecture consists of p processes, each with its own exclusive address space. Our message-passing programs are based on the Single Program Multiple Data (SPMD) paradigm [137] and follow a loosely synchronous approach: all processes execute the same code, containing mainly asynchronous tasks, but also a few tasks that synchronize to perform interactions.

5.1.1 OR Parallel RRT

The simplest way to parallelize RRT is to apply the OR parallel paradigm. Algorithm 16 shows the *OR parallel* RRT, as defined in [25]. Each process performs its own RRT (lines 1-8), and the first one to reach a stopping criterion broadcasts a termination message and returns its solution tree (lines 9-11). This broadcast operation cannot actually be implemented as a regular MPI.Broadcast routine, as this collective operation would require all processes to synchronize. Rather, the first process to finish sends a termination message to all the others, using MPI.Send routines matched with MPI.Receive routines. As it is not known beforehand when these interactions should happen, a non-blocking receiving operation that will “catch” the termination message is initiated before entering the **while** loop. The **received** (endMsg) operation is implemented as an MPI.Test routine checking the status (completed or pending)

¹<http://www.mpi-forum.org>

Algorithm 17: Distributed RRT

```

input : the feasible path planning problem  $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$ 
output: the tree  $\mathcal{T}$ 
1  $\mathcal{T} \leftarrow \text{initTree}(q_{\text{init}})$ 
2 while not ( $\text{stoppingCriteria}(\mathcal{T})$  or  $\text{received}(\text{endMsg})$ ) do
3   while  $\text{received}(\text{nodeData}(q_{\text{new}}, q_{\text{near}}))$  do
4      $\text{addNewNode}(\mathcal{T}, q_{\text{new}})$ 
5      $\text{addNewEdge}(\mathcal{T}, q_{\text{near}}, q_{\text{new}})$ 
6    $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
7    $q_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}, q_{\text{rand}})$ 
8    $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}, q_{\text{rand}})$ 
9   if  $q_{\text{new}} \neq \text{null}$  then
10     $\text{addNewNode}(\mathcal{T}, q_{\text{new}})$ 
11     $\text{addNewEdge}(\mathcal{T}, q_{\text{near}}, q_{\text{new}})$ 
12     $\text{broadcast}(\text{nodeData}(q_{\text{new}}, q_{\text{near}}))$ 
13 if  $\text{stoppingCriteria}(\mathcal{T})$  then
14    $\text{broadcast}(\text{endMsg})$ 
15   return  $\mathcal{T}$ 

```

of the request generated by the non-blocking receiving operation. Finally, in case of several processes reaching a solution at the same time, the program ends with a collective operation for them to synchronize and agree on which one should report its solution. Note that communications are negligible in the total runtime.

5.1.2 Collaborative Building of a Single RRT

Instead of constructing several RRTs concurrently, another possibility is to have all processes collaborating to build a single RRT. Parallelization is then achieved by partitioning the building task into sub-tasks assigned to the processes. We propose two ways of doing so, based on classical decomposition techniques. 1) Since constructing an RRT consists of exploring a search space, we can use an *exploratory decomposition* [72]. Each process performs its own sampling of the space (but without any space partitioning involved) and maintains its own copy of the tree, exchanging new nodes with other processes. This leads to a distributed (or decentralized) scheme where no task scheduling is required, aside from a termination detection mechanism. 2) Another classical approach is to perform a *functional decomposition* of the task [64]. In the RRT algorithm, two kinds of sub-tasks can be distinguished: the ones that require to access the tree (initializing it, adding new nodes and edges, finding the nearest neighbor of q_{rand} , and evaluating the stopping criteria) and those that do not (sampling a random configuration and performing the extension step). This leads to the choice of a manager-worker (or master-slave) scheme as the dynamic and centralized task-scheduling strategy: the manager maintains the tree, and the workers perform the actions requiring no access to it.

Distributed RRT

The *Distributed* RRT is presented in Algorithm 17. In each iteration of the tree construction loop (lines 2-12), each process first checks whether it has received new nodes from other processes (line 3) and, if so, adds them to its local copy of the tree (lines 4-5). Then, each process performs an expansion attempt (lines 6-8). If it succeeds (line 9), the process adds the new node to its local tree (lines 10-11) and broadcasts the node (line 12). Adding all the received nodes before attempting an expansion ensures that every process works with the

Algorithm 18: Manager-worker RRT

```

input : the feasible path planning problem  $(\mathcal{C}, q_{\text{init}}, q_{\text{goal}})$ 
output: the tree  $\mathcal{T}$ 
1 if  $processID = mgr$  then
2    $\mathcal{T} \leftarrow \text{initTree}(q_{\text{init}})$ 
3   while not  $\text{stoppingCriteria}(\mathcal{T})$  do
4     while  $\text{received}(\text{nodeData}(q_{\text{new}}, q_{\text{near}}))$  do
5        $\text{addNewNode}(\mathcal{T}, q_{\text{new}})$ 
6        $\text{addNewEdge}(\mathcal{T}, q_{\text{near}}, q_{\text{new}})$ 
7        $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
8        $q_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}, q_{\text{rand}})$ 
9        $w \leftarrow \text{chooseWorker}()$ 
10       $\text{send}(\text{expansionData}(q_{\text{rand}}, q_{\text{near}}), w)$ 
11     $\text{broadcast}(\text{endMsg})$ 
12    return  $\mathcal{T}$ 
13 else
14   while not  $\text{received}(\text{endMsg})$  do
15      $\text{receive}(\text{expansionData}(q_{\text{rand}}, q_{\text{near}}), mgr)$ 
16      $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}, q_{\text{rand}})$ 
17     if  $q_{\text{new}} \neq \text{null}$  then
18        $\text{send}(\text{nodeData}(q_{\text{new}}, q_{\text{near}}), mgr)$ 

```

most up-to-date state of the tree. Note that processes never wait for messages; they simply process them as they arrive. At the end, the first process to reach a stopping criterion broadcasts a termination message and returns its tree (lines 13-15). This broadcast operation is implemented in the same way as for the OR parallel RRT. In a similar manner, the broadcast of new nodes (line 12) is not implemented as a regular MPI.Broadcast routine, which would cause all processes to wait for each other. As a classical way to overlap computation with interactions, we again use MPI.Send routines matched with non-blocking MPI.Receive routines. That way, the `received(nodeData)` test (line 3) is performed by checking the status of the request associated with a non-blocking receiving operation initiated beforehand, the first one being triggered before entering the **while** loop, and the subsequent ones being triggered each time a new node is received and processed. Again, we have to deal with the case of several processes reaching a solution at the same time. Finally, a Universally Unique Identifier (UUID) is associated with each node to provide processes with a homogeneous way of referring to the nodes. UUIDs are classically used in distributed environments, but, as an alternative, we could have used the pair $(processID, nodeID_forThisProcess)$.

Manager-Worker RRT

The *Manager-worker* RRT is introduced in Algorithm 18. It contains both the code executed by the manager (lines 2-12) and the code executed by the workers (lines 14-18). The manager is the only process that can access the tree, and it delegates the expansion attempts to the workers. The expansion is generally the most computationally-expensive stage in the tree construction because it involves motion simulation and validation. The manager could also delegate the sampling step, but this would be worthless because of the low computational cost of this operation in our settings (i.e. in the standard case of a uniform random sampling in the whole configuration space): the additional communication cost would outweigh any potential benefit.

At each iteration of the construction loop (lines 3-10) the manager first checks whether it has received new nodes from the workers (line 4). If so, it adds them to the tree (lines 5-6). Then, it samples a random configuration (line 7) and identifies its closest neighbor in the tree (line 8). Next, it looks for an idle worker (line 9), which means potentially going through a waiting phase, and sends the data needed to perform an expansion attempt to the worker (line 10). Finally, when a stopping criterion is reached, it broadcasts a termination message (line 11). Workers remain active until they receive this message (line 14), but they can go through waiting phases. During each computing phase, a worker receives some data from the manager (line 15) and performs an expansion attempt (line 16). If it succeeds (line 17), it sends the new node to the manager (line 18).

Contrary to the previous ones, this algorithm does not require non-blocking receiving operations to broadcast the termination message. Workers being idle if they receive no data from the manager, there is no need to overlap computation and interactions. Before entering a computing phase, a worker waits on a blocking `MPI_Receive` routine implementing both the `receive(expansionData)` operation and the `received(endMsg)` test. The type of received message determines its next action: stopping or attempting an expansion. On the manager side, blocking `MPI_Send` routines implement the `broadcast(endMsg)` and `send(expansionData)` operations. The remaining question about the latter is: to which worker should the data be sent. An important task of the manager is to perform load-balancing among workers through the `chooseWorker()` function. For that, it keeps track of the status (busy or idle) of all workers and sends one sub-task at a time to an idle worker, choosing it in a round robin fashion. If all workers are busy, the manager waits until it receives a message from one of them, that then becomes idle. This has two consequences. First, on the worker side, the `send(nodeData)` operation covers two `MPI_Send` routines: one invoked to send new nodes when the expansion attempt succeeds, and the other containing no data, used otherwise. Second, on the manager side, two matching receiving operations are implemented via non-blocking `MPI_Receive` routines, allowing to use `MPI_Wait` routines if necessary. This also enables to implement the `received(nodeData)` test with an `MPI_Test` routine. These non-blocking receiving operations are initiated before entering the `while` loop, and re-initiated each time the manager receives and processes a message. Finally, to reduce the communication costs of the `send(nodeData)` operation, workers do not send back the configuration q_{near} . Rather, the manager keeps track of the data it sends to workers, thus avoiding the need for UUIDs.

5.1.3 Implementation Framework

Since the sequential implementation of RRT we wanted to parallelize was written in C++, and MPI being targeted at C and Fortran, we had to use a C++ binding of MPI. We were also confronted with the low-level way in which MPI deals with communications, requiring the programmer to explicitly specify the size of each message. In our application, messages were to contain instances of high-level classes, whose attributes could be pointers or STL containers. Thus, we decided to exploit the higher-level abstraction provided by the Boost.MPI library². Coupled with the Boost.Serialization library³, it enables processes to easily exchange class instances, making the tasks of gathering, packing and unpacking the underlying data transparent to the programmer. We also used the implementation of UUIDs⁴ provided by the Boost library.

²<http://www.boost.org/doc/libs/1.47.0/doc/html/mpi.html>

³<http://www.boost.org/doc/libs/1.47.0/libs/serialization/doc/index.html>

⁴<http://www.boost.org/doc/libs/1.47.0/libs/uuid/index.html>

5.2 Experiments with RRT and T-RRT

Before presenting the results of the experiments, we introduce the metrics used to evaluate the parallel algorithms. We also present the parallel platform we have worked on, and the motion planning problems we have studied. We then explain the two experiments we have performed, and report general results. A detailed analysis of the performance of the algorithms will be the focus of Section 5.3.

5.2.1 Path Planning Problems and Evaluation Settings

Aimed at assessing the performance gain achieved by a parallel algorithm run on p processors, the *speedup* S is defined as the ratio of the sequential runtime to the parallel runtime: $S(p) = T_S / T_P(p)$ [64, 72]. The parallel runtime $T_P(p)$ is measured on a parallel computer, using p processors, and the sequential runtime T_S is measured on a single processor of this computer. We define $T_P(p)$ (resp. T_S) as the mean time needed to reach a solution, by averaging the runtimes obtained over 100 executions of a parallel (resp. sequential) algorithm. Another common metric we use is the *efficiency* E of a parallel algorithm, which is defined as the ratio of the speedup to the number of processors: $E(p) = S(p) / p$ [64, 72].

The numerical results presented in this paper have been obtained by running the algorithms on MareNostrum, the parallel platform of the Barcelona Supercomputing Center. It is an IBM cluster platform composed of 2560 IBM BladeCenter JS21 blade servers connected by a Myrinet local area network warranting 2 Gbit/s of bandwidth. Each server includes two 64-bit dual-core PowerPC 970MP processors at 2.3 GHz, sharing 8 GB of memory. The implementation of MPI installed on this platform is MPICH2⁵.

We have evaluated the parallel algorithms on three path planning problems involving molecular models, using the molecular motion planning toolkit we are currently developing [39]. The three problems involve free-flying objects (i.e. 6 DoFs)⁶. They are characterized by different configuration-space topologies (cf. Fig. 5.1). *Passage* is a protein-ligand exit problem: a ligand exits the active site of a protein through a relatively short and large pathway locally constrained by several side-chains. *Corridor* is a similar problem, but with a longer and narrower exit pathway, i.e. more geometrically constrained than *Passage*. In *Roundabout*, a protein goes around another one in an empty space, thus involving the weakest geometrical constraints, but the longest distance to cover. For more details on these examples the interested reader is referred to [39].

5.2.2 First Experiment - High Expansion Cost

Our first experiment aims at assessing the speedup achieved by the parallel variants of an RRT-like algorithm. More precisely, this experiment involves parallel versions of T-RRT in its Extend version. In this context, the expansion cost is dominated by the energy evaluation of molecular motions, which is much more costly than the simple collision detection performed by RRT. This exemplifies the case where the computational cost of the RRT expansion is significantly greater than communication cost. This is a favorable situation for an MPI-based parallelization (as illustrated by the results of the second experiment) because the communication overhead is outweighed by the sharing of high-cost workload-units between processes [64]. Such a situation happens when planning motions of complex systems, whether molecules or robots (as discussed in Section 5.2.4).

⁵<http://www.mpich.org>

⁶Having a common dimensionality across examples facilitates the evaluation of the algorithms. Increasing dimensionality would mainly raise the computational cost of the nearest neighbor search. Note that, however, this cost becomes almost dimension-independent when using projections on a lower-dimensional space, without a significant loss in accuracy [136].

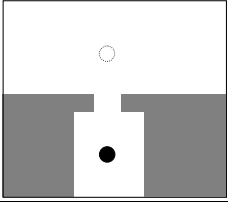
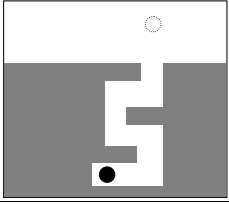
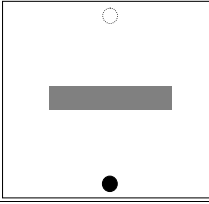
Problem name		<i>Passage</i>	<i>Corridor</i>	<i>Roundabout</i>
Problem type				
Sequential T-RRT	T_S (s)	48 ± 18	1250 ± 1000	38 ± 18
	N_S	644 ± 119	1030 ± 695	655 ± 336
	X_S	1810 ± 690	45400 ± 40900	1130 ± 456

Figure 5.1: Schematic representation of the configuration spaces of the three path-planning problems, and results obtained with the sequential T-RRT (that involves molecular energy computation). Average values over 100 runs (and standard deviation) are given for the sequential runtime, T_S (in seconds), the number of nodes in the tree, N_S , and the number of expansion attempts, X_S .

Fig. 5.1 presents the numerical results obtained with the sequential version of T-RRT. Fig. 5.2 shows the speedup achieved by the parallel algorithms on each problem. The OR parallel T-RRT always shows a poor speedup, but the speedup achieved by the Distributed T-RRT and the Manager-worker T-RRT can be really high. Differences between problems are significant, the best speedup being achieved on the most constrained problem, *Corridor*, then *Passage*, then *Roundabout*. These results are further explained in the analysis presented in Section 5.3.

5.2.3 Second Experiment - Variable Expansion Cost

In our second experiment, we study how the speedup achieved by the parallel algorithms evolves in relation to the computational cost of an RRT expansion. In parallel programming, speedup generally improves as the computational cost of a process workload-unit increases with respect to the communication overhead [64]. To test that, we run a controlled experiment in which we artificially increase the cost of the RRT expansion. For that, we use RRT in its Extend version, where motion validation is reduced to collision detection (meaning that no energy evaluation is involved), thus performing low-cost expansions. To increase the expansion cost c , we repeat t times the collision detection test in the `extend()` function. Note that we estimate c by dividing the sequential runtime by the number of expansion attempts. Finally, c is varied by varying t .

Fig. 5.3 shows how the speedup and efficiency achieved by the parallel algorithms vary with respect to the expansion cost c , when run on 32 processors. As the number of processors is fixed, efficiency is proportional to speedup. The speedup of the OR parallel RRT does not change with c . In other words, it is not influenced by the ratio between computation and communication costs. On the other hand, this ratio strongly impacts the speedup of the Distributed and Manager-worker RRT. They both achieve a very low speedup when c is low: the first point of each curve, obtained with $t = 1$, shows that in this case the parallel version is even slower than the sequential one (i.e. $S < 1$). When c increases, both algorithms show a similar and important increase in speedup. The magnitude of this increase is strongly influenced by the studied problem: it is the greatest on the most constrained problem, *Corridor* (for which almost optimal efficiency is achieved), then *Passage*, then *Roundabout*. When c is high, making communication load insignificant compared to computation load, the speedup reaches a plateau.

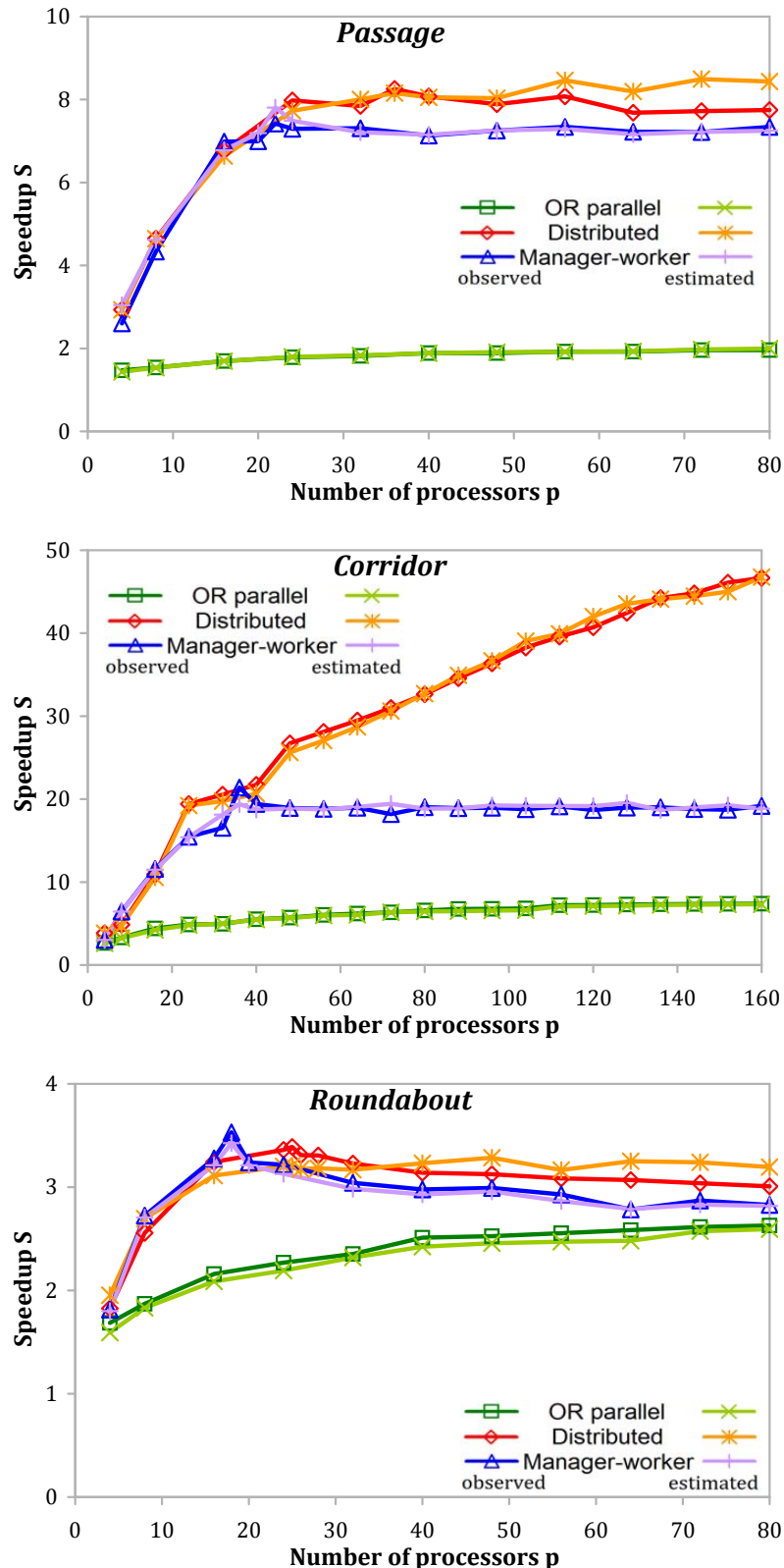


Figure 5.2: Speedup (averaged over 100 runs) achieved by the parallel algorithms in relation to the number of processors, on the *Passage*, *Corridor* and *Roundabout* problems (first experiment). Both the observed speedup and the speedup estimated by the models presented in Section 5.3 are reported.

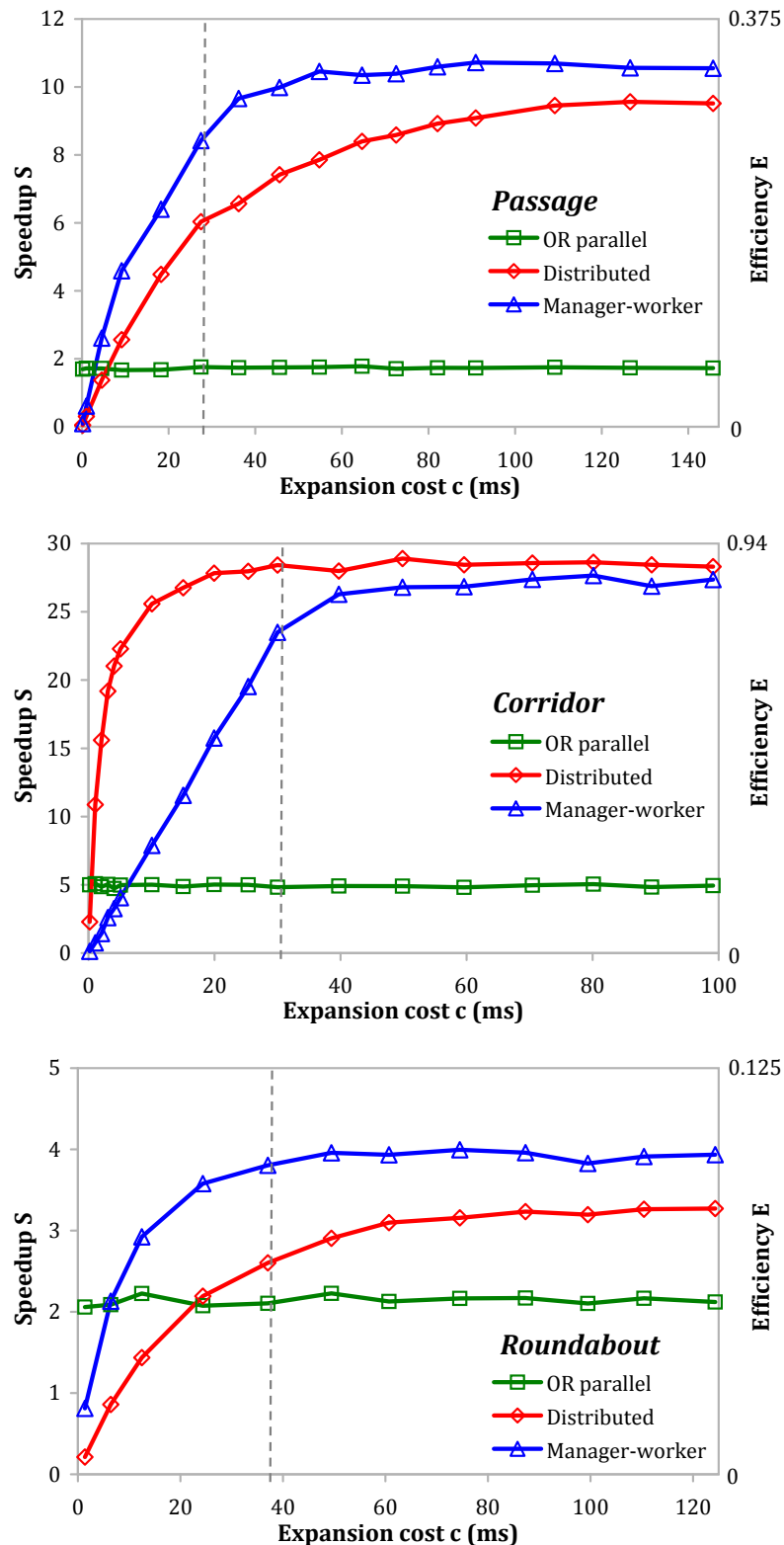


Figure 5.3: Speedup (left axis) and efficiency (right axis), averaged over 100 runs, achieved by the parallel algorithms, in relation to the computational cost of the RRT expansion (in milliseconds), when solving the *Passage*, *Corridor* and *Roundabout* problems on 32 processors (second experiment). As a reference, the dashed vertical line shows the cost of the T-RRT expansion (as estimated in the first experiment).

5.2.4 Robotic Examples

Our results show that the Distributed and Manager-worker RRT are beneficial on problems for which the computational cost of an RRT expansion c is significantly greater than the cost of a communication. The communication cost being about 1 ms on MareNostrum, we obtain a good speedup when c is greater than 25 ms (cf. Fig. 5.3). This means that most academic path-planning benchmarks cannot benefit from an MPI-based parallelization of RRT-like algorithms. Indeed, these examples often reduce motion validation to collision detection in geometrically-simple scenes, leading to a fast RRT expansion. However, in the context of robot path planning, high-cost expansions may occur in various situations. The first one is the case of high geometric complexity, when objects of the world are represented by large numbers of polyhedral faces. For example, c is about 27 ms on the *flange* benchmark [139], and about 28 ms on the *exhaust disassembly* problem [41], despite efficient collision detection. High-cost expansions may also occur on problems under kinodynamic constraints requiring to use a dynamic simulator [40]. Another case is when planning on constraint manifolds embedded in higher-dimensional ambient spaces [13], especially with complex systems such as closed-chain mechanisms. For example, c is about 120 ms on a problem where the *Justin* robot transports a tray in a cluttered environment [70]. An even more complex case is task-based path planning involving humanoid robots with dynamic constraints [22,42]. For example, c is greater than 1 s on a problem where two *HRP-2* robots collaboratively transport a table [22]. Due to their high expansion costs, all these examples would benefit from a similar or even higher speedup than those we have studied. This illustrates that a large class of practical problems involving complex environments and complex robot systems can benefit from an MPI-based parallelization of RRT-like algorithms.

5.3 Analysis of the Parallel Algorithms

The experiments we have presented in the previous section provide the first clues on the differences in behavior between the parallel versions of RRT. Nevertheless, the resulting speedup curves are not sufficient to understand performance variations due to the problem type, the number of processors involved or the computational cost of the RRT expansion. This is what we analyze now for each parallel algorithm.

5.3.1 OR Parallel RRT

The OR parallel RRT does not rely on sharing the computation load among processes, but on finding small-sized solutions that are faster to compute. The more processes are involved, the greater is the chance to find a solution quickly. On average, the number of expansions attempted by the OR parallel RRT on p processors, $X_P(p)$, decreases with p . Similarly, the number of tree nodes, $N_P(p)$, decreases with p (cf. Fig. 5.4). If we express the parallel runtime as $T_P(p) = X_P(p) \cdot c$, where c is the expansion cost, we get that $T_P(p)$ decreases with p . If the sequential runtime is similarly expressed as $T_S = X_S \cdot c$, with X_S the number of expansions attempted by the sequential RRT, we have:

$$S(p) = \frac{X_S}{X_P(p)} \quad (5.1)$$

Fig. 5.2 illustrates the evolution with respect to p of both the observed speedup (computed using runtimes averaged over 100 runs) and the speedup estimated by (5.1) (computed using values of X_S and $X_P(p)$ averaged over 100 runs). The graphs show that the estimated speedup values fit well the observed data. Important features of the behavior of the OR parallel RRT are reflected in (5.1). First, S is independent from the expansion cost c because

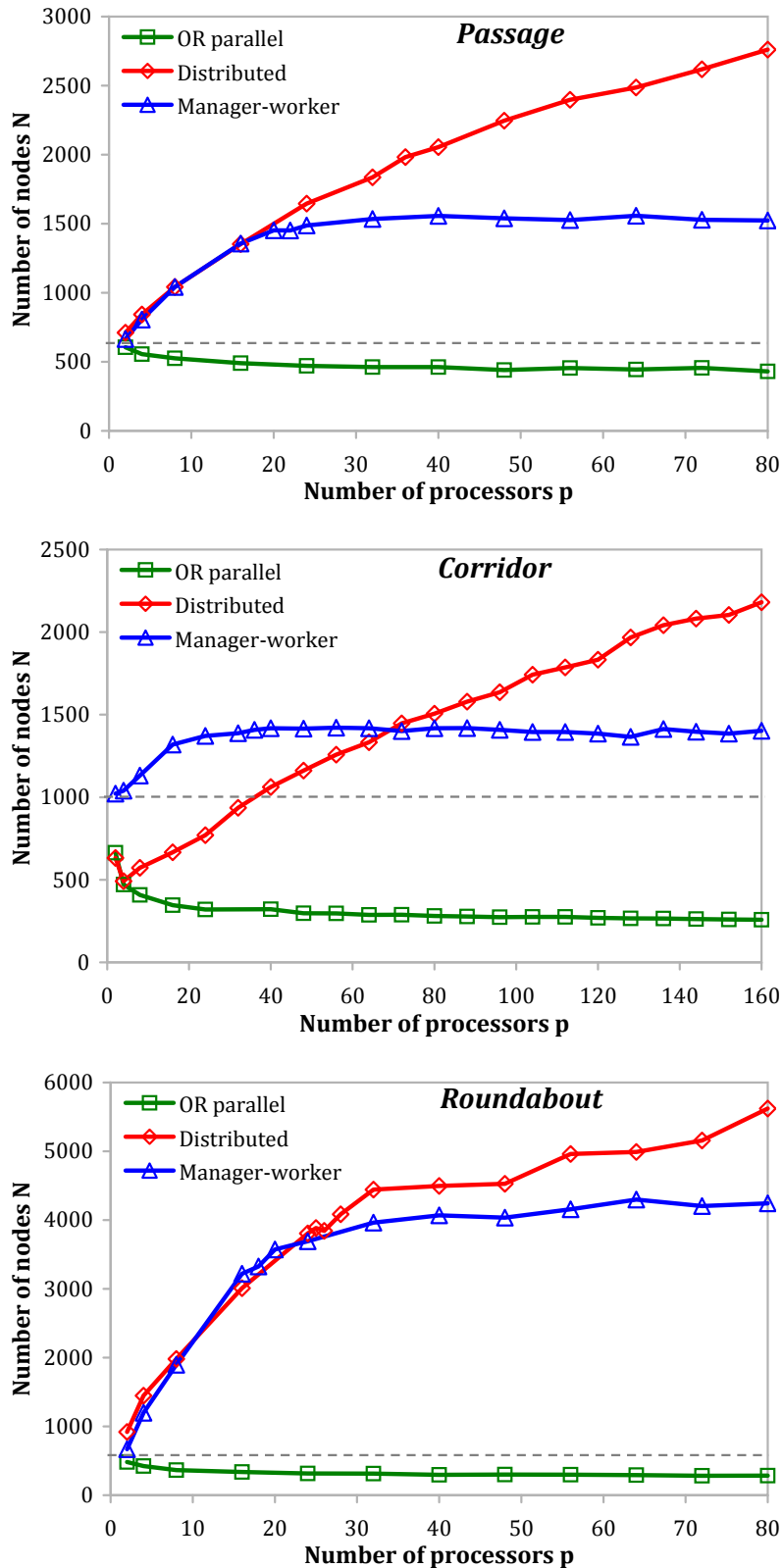


Figure 5.4: Number of nodes (averaged over 100 runs) in the trees produced by the parallel algorithms in relation to the number of processors, on the *Passage*, *Corridor* and *Roundabout* problems (first experiment). The dashed horizontal line shows the number of nodes in the trees generated by the sequential T-RRT.

Table 5.1: Summary of the numerical results obtained with the sequential and parallel algorithms on the MareNostrum platform.

		<i>Passage</i>	<i>Corridor</i>	<i>Roundabout</i>
sequential runtime variability		0.4	0.8	0.5
OR parallel RRT	S_{\max}	2	8	2.7
Distributed RRT	\bar{p}	36	> 160	25
	S_{\max}	8.3	> 50	3.4
Manager-worker RRT	\bar{p}	22	36	18
	S_{\max}	7.8	21.4	3.5

X is independent from it. This confirms what we could deduce from the fact that the efficiency curves of the OR parallel RRT are almost flat (cf. Fig. 5.3). Second, the only factor influencing the evolution of $S(p)$ is $X_P(p)$, which decreases with p and is lower-bounded by the minimum number of expansion attempts required to reach a solution. This explains why $S(p)$ increases with p toward an asymptotic value S_{\max} (equal to 2, 8 and 2.7 on *Passage*, *Corridor* and *Roundabout* respectively, as shown by Fig. 5.2). If we define the variability in sequential runtime by the ratio of the standard deviation to the mean of the runtime T_S reported in Fig. 5.1, we get the values 0.4, 0.8 and 0.5 for *Passage*, *Corridor* and *Roundabout* respectively. Table 5.1 shows that S_{\max} is strongly positively correlated with this sequential runtime variability.

5.3.2 Distributed RRT

In the Distributed RRT, the computation load is shared among processes. It can again be expressed as $X_P(p) \cdot c$, where $X_P(p)$ decreases with p thanks to work sharing. A significant communication load is added to the global workload, but communications happen only after a new node is built. If we assume the tree construction is equally shared among processes, from the $N_P(p)$ tree nodes, each process will have contributed $N_P(p) / p$. Furthermore, each process sends this amount of nodes to, and receives this amount of nodes from, each of the $p-1$ other processes. The total communication load between all processes can thus be estimated by $2(p-1) \cdot (N_P(p) / p) \cdot m$, where m is the cost of sending one node between two processes. Therefore, we have: $T_P(p) = X_P(p) \cdot c + \frac{2(p-1)}{p} \cdot N_P(p) \cdot m$. This highlights the fact that the workload repartition between computation and communication mainly depends on the ratio $\frac{c}{m}$. Finally, we get:

$$S(p) = \frac{X_S \cdot c}{X_P(p) \cdot c + \frac{2(p-1)}{p} \cdot N_P(p) \cdot m} \quad (5.2)$$

Fig. 5.2 illustrates the evolution with respect to p of both the observed speedup and the speedup estimated by (5.2) (and computed using numbers of nodes and expansion attempts averaged over 100 runs). Knowing that $T_P(2) = X_P(2) \cdot c + N_P(2) \cdot m$, we can estimate m by running the Distributed RRT on two processors. The graphs show that the estimated speedup provides a good fit to the observed speedup. The main factor allowing $S(p)$ to increase with p is work sharing, i.e. the decrease of $X_P(p)$. Another beneficial factor is what we call the ‘‘OR parallel effect’’: as each process performs its own sampling of the search space, when few processes are involved, the Distributed RRT reaches smaller solutions than the sequential RRT. Fig. 5.4 shows that this happens mainly on problems whose sequential runtime variability is high, such as *Corridor*: in the middle graph, the curve representing $N_P(p)$ for the Distributed RRT is below the horizontal line representing N_S when p is low. On the other hand, an important factor hampers the increase in speedup. When collaboratively building an RRT, a side-effect of adding more processes is to change the balance between exploration

and refinement (these terms being used as in [84]) in favor of refinement. Therefore, more expansions are attempted globally (i.e. $p \cdot X_P(p)$ increases with p) and larger trees are produced (i.e. $N_P(p)$ increases with p , as shown by Fig. 5.4). As a result, the overall computation load increases with p .

The denominator of (5.2) represents the workload of a single process. Even though the global computation load for all processes increases with p , the computation load for one process, $X_P(p) \cdot c$, decreases with p . However, the communication load for one process, $\frac{2(p-1)}{p} \cdot N_P(p) \cdot m$, increases with p because $N_P(p)$ increases with p and $\frac{2(p-1)}{p}$ increases with p within $[1, 2)$. The decrease in computation load seems to dominate, since Fig. 5.2 mainly shows an increase in speedup for the Distributed RRT. However, it appears from the least constrained problem, *Roundabout*, that when p becomes too high the speedup decreases slightly. The optimal observed speedup S_{\max} is 8.3 and 3.4 for *Passage* and *Roundabout*, and seems to be greater than 50 for *Corridor* (cf. Fig. 5.2). It is achieved for an optimal value of p , denoted by \bar{p} , equal to 36 and 25 for *Passage* and *Roundabout*, and greater than 160 for *Corridor* (cf. Fig. 5.2). Table 5.1 shows that \bar{p} and S_{\max} are strongly positively correlated: the more processes can collaborate without increasing refinement too much, the higher S_{\max} is. The increase in refinement is observed through the increase in number of nodes (cf. Fig. 5.4). It appears that problems characterized by weak geometrical constraints, such as *Roundabout*, are more sensitive to this issue, leading to poor speedup. For problems characterized by strong geometrical constraints, such as *Corridor*, the speedup scales better with respect to the expansion cost c (cf. Fig. 5.3).

5.3.3 Manager-Worker RRT

In the Manager-worker RRT, each expansion attempt is preceded by a communication from the manager to a worker, and each successful expansion is followed by a communication from a worker to the manager. Being empty, the message sent after a failed expansion can be ignored. In the trivial case of the manager using a single worker, communication and computation cannot overlap, and thus $T_P(2) = X_P(2) \cdot c + (X_P(2) + N_P(2)) \cdot m$, where m is the cost of sending a message. We estimate m by running tests on two processors and using this formula. If more workers are available, two cases should be considered. First, if communication is more costly than computation (i.e. $m > c$) the manager can use at most two workers at a time: while it sends some data to a worker, the other worker has already finished its computation. In that case, we have $T_P(p) = (X_P(p) + N_P(p)) \cdot m > T_S$, and parallelization is useless. Second, if $c > m$, more than two workers can be used, but the manager is still a potential bottleneck depending on the ratio $\frac{c}{m}$: the less significant the communication cost is compared to the expansion cost, the more workers can be used. For given values of c and m , at most \bar{p} processors can be used, and thus, the number of workers effectively used is $\min(p-1, \bar{p}-1)$. Assuming the computation load is equally shared among workers, we have:

$$S(p) = \frac{X_S \cdot c}{\frac{X_P(p)}{\min(p-1, \bar{p}-1)} \cdot c + (X_P(p) + N_P(p)) \cdot m} \quad (5.3)$$

The speedup estimated by (5.3) shows a good fit to the observed speedup of the Manager-worker RRT (cf. Fig. 5.2). Equation (5.3) explains how the speedup evolves with respect to p and c . When $p \leq \bar{p}$, $S(p)$ increases with p thanks to work sharing among workers. However, when $p > \bar{p}$, increasing p becomes useless. Therefore, $S(p)$ reaches a plateau around a value S_{\max} equal to 7.8, 21.4 and 3.5 for *Passage*, *Corridor* and *Roundabout* (cf. Fig. 5.2). In fact, \bar{p} is the value of p for which $S(p)$ reaches S_{\max} : it is equal to 22, 36 and 18 for *Passage*, *Corridor* and *Roundabout* respectively (cf. Fig. 5.2). Obviously, S_{\max} is strongly positively correlated with \bar{p} (cf. Table 5.1). Moreover, the second experiment shows that \bar{p} increases with c . This explains why we observe on Fig. 5.3 that S increases with c at first, and then reaches

Table 5.2: Summary of the numerical results obtained with the parallel algorithms on the Cacao platform.

		<i>Passage</i>	<i>Corridor</i>	<i>Roundabout</i>
OR parallel RRT	S_{\max}	2	8	2.4
Distributed RRT	\bar{p}	22	> 160	28
	S_{\max}	6.3	> 65	2.7
Manager-worker RRT	\bar{p}	25	31	23
	S_{\max}	8.8	23.5	3.2

a plateau: when \bar{p} reaches 32 (the number of processors used in the second experiment), S cannot increase anymore. Contrary to the Distributed RRT, the Manager-worker RRT does not benefit from the “OR parallel effect”: in Fig. 5.4, the curve of $N_P(p)$ is never below the horizontal line representing N_S . As a consequence, the Manager-worker RRT shows a lower speedup than the Distributed RRT on problems with a high variability in sequential runtime, such as *Corridor* (cf. Fig. 5.2). Besides, it suffers from the increase in refinement, which translates into $X_P(p)$ and $N_P(p)$ increasing with p , when $p \leq \bar{p}$ (cf. Fig. 5.4). Path-planning problems characterized by weak geometrical constraints, such as *Roundabout*, are more sensitive to this issue.

5.3.4 Discussion

To evaluate the influence of the architecture, we have performed the two previous experiments on another parallel platform, Cacao, available in our laboratory. Cacao is a small cluster composed of 24 HP ProLiant DL160 G5 servers connected by a high-speed InfiniBand switch warranting 10 Gbit/s of bandwidth, and using OpenMPI⁷. Each server includes two 64-bit quad-core Intel Xeon E5430 processors at 2.66 GHz, with 12 MB of L2 cache, and sharing 7.79 GB of memory. We aimed to assess 1) the consistency of performance of the parallel algorithms and 2) the goodness-of-fit of the models provided by (5.1), (5.2) and (5.3). First, we observe that the models are robust and provide good estimations of the speedup achieved on Cacao. Second, the numerical results obtained on Cacao and reported in Table 5.2 are similar to those obtained on MareNostrum (cf. Table 5.1). The speedup of the OR parallel RRT is the same on both architectures because no communication is involved. The Distributed RRT is more impacted than the Manager-worker RRT by the choice of the architecture because its “ n to n ” communication scheme makes it more sensitive to the level of optimization of the MPI communications. As a result, when communications are less efficient (as observed on Cacao) the Distributed RRT can be outperformed by the Manager-worker RRT on less constrained problems (such as *Passage* and *Roundabout*) characterized by a low variability in sequential runtime.

One may wonder whether the Manager-worker RRT could be improved by assigning workers batches of multiple expansion attempts instead of single ones. Even though it reduces communications, after evaluation this idea appears to yield mixed results. The drawback of this variant is to further worsen the main hindrance affecting the Manager-worker RRT, namely the increase in refinement with respect to p . If k is the size of a batch of expansion attempts, we observe that X_P and N_P increase with k . On problems for which the success rate of an RRT expansion is high ($N/X = 1/3$ for *Passage* and $1/2$ for *Roundabout*), using this modification reduces speedup, even with low values of k . Nevertheless, speedup increases slightly on the *Corridor* problem, where this success rate is much lower ($N/X = 1/50$), except when k becomes too high.

⁷<http://www.open-mpi.org>

5.4 Application to Other RRT-like Algorithms

As already mentioned, some extensions of RRT can be parallelized using the exact same schemes as those presented in Algorithms 16, 17 and 18. This is what we have done for the Extend version of T-RRT in Section 5.2, and it could have been done for other mono-directional versions of T-RRT, such as the Connect or Goal-biased versions (cf. Section 3.1). We could also develop a straightforward parallelization of the Anytime T-RRT (cf. Section 4.1). In the Distributed case, this would require processes to exchange information about the added edges. In the Manager-worker case, the manager would have to perform the cycle-addition procedure, which would increase its workload.

Sampling-based tree planners similar to RRT, such as RRT* [93] or EST [75], can also benefit from this work. For example, EST can be parallelized exactly in the same way as RRT because its `propagate` function is the exact counterpart of the `extend` function of RRT. On the other hand, parallelizing RRT* would be much more involved, except for the OR parallel version. Besides new vertices, messages exchanged between processes should include added and removed edges, thus increasing the communication load. This could be balanced in the Distributed version by the higher cost of the expansion in RRT* than in RRT. However, as one RRT* expansion intertwines operations requiring or not access to the tree, a Manager-worker version of RRT* would not be very efficient. Furthermore, the Transition-based RRT* (cf. Section 4.1) would suffer from the same issues.

Path planning algorithms building several trees can also benefit from this work. For example, in the bidirectional-RRT variant where both trees are extended toward the same random configuration [106], processes can be separated in two groups applying our parallel algorithms, and getting random configurations from an extra process. More generally, any multiple-tree variant of EST, RRT or T-RRT (cf. Sections 3.2 and 3.3) could be parallelized similarly to what is proposed in [134], where the focus is on distributing the operations that allow connecting trees. It would be interesting to combine the underlying principles of our work and the work in [134]. This would provide two complementary “layers” of parallelization: the distribution of the construction of a single tree, and the distribution of connection attempts between trees. Seen from this perspective, the work in [134] is an extension of the OR parallel RRT because each tree is built by a single process. In addition, it would be interesting to analyze similar extensions of the Distributed and Manager-worker RRT.

5.5 Conclusion

In this chapter, we have presented and analyzed three parallel versions of RRT-like algorithms, designed for distributed-memory architectures, and using MPI: OR parallel RRT, Distributed RRT and Manager-worker RRT. The OR parallel RRT was first introduced in [25] and reused on shared memory in [2]. The Distributed RRT and Manager-worker RRT are the counterparts for distributed memory of the AND (or embarrassingly parallel) RRT used on shared memory [2, 25]. We have shown that parallelizing RRT-like algorithms with MPI can provide substantial performance improvement in two cases. First, problems whose variability in sequential runtime is high can benefit from the OR parallel RRT. Second, problems for which the computational cost of an RRT expansion is high can benefit from the Distributed RRT and Manager-worker RRT.

The empirical results and the performance analysis reveal that the best parallelization scheme depends on the studied problem, the computational cost of an RRT expansion, and the parallel architecture. The Distributed and Manager-worker RRT provide a good speedup, except on problems with weak geometrical constraints. In that case, they suffer from an increase in refinement (vs. exploration) translating into greater overall computation and communication loads. On problems showing a low variability in sequential runtime, depending

on the architecture, the Manager-worker RRT can outperform the Distributed RRT. On the other hand, if the sequential runtime variability is high, the Distributed RRT outperforms the Manager-worker RRT thanks to its “OR parallel effect”.

Based on these results, and as future work, we plan to improve the three parallelization schemes presented here. First, note that the Distributed RRT can suffer from memory-overhead issues because each process maintains its own tree. To address this, we plan to better exploit the architecture of cluster platforms by combining message passing with multi-threading and allowing the processes sharing the same memory to build a common tree. Second, in the Manager-worker RRT, to avoid seeing the manager becoming a bottleneck, a hierarchical approach involving several managers can be developed. Third, we plan to investigate approaches combining several of the three paradigms. For example, integrating the OR parallel RRT into the Manager-worker RRT could allow it to perform better on problems showing a high variability in sequential runtime. Finally, instead of parallelizing RRT itself, we could also parallelize its most computationally expensive components, such as the collision detection, as done in [15].

Another interesting direction for research is to intertwine different levels of parallelization in our methods. This would be extremely beneficial to the multiple-tree variant of T-RRT that we use in Chapters 6 and 7. In this context, we could combine three levels of parallelization: 1) distributing the construction of the trees over several groups of processes; 2) sharing the construction of each tree between several processes; 3) parallelizing the most computationally-expensive components of the T-RRT expansion. Using such parallel version of the Multi-T-RRT would allow solving very complex problems in robotics and structural biology.

Chapter 6

Application to Complex Robotic Problems

In this chapter, we present two applications of the algorithms discussed in this thesis, within the field of robotics. We show that, by developing sophisticated continuous configuration-cost functions that are more application-specific than the basic clearance-based cost function used in academic examples, it is possible to deal with complex realistic problems. Indeed, such cost functions can take into account the constraints inherent to complex robotic systems, such as aerial towed-cable systems involving several flying robots. We also show that, by combining various of the extensions of T-RRT we have introduced in this thesis, it is possible to address path-planning problems that are more challenging than traditional “init-to-goal” problems, such as ordering-and-pathfinding problems.

First, we propose an innovative path planning approach for reliable, 6-dimensional, quasi-static manipulation with an aerial towed-cable system. The novelty of this approach lies in the use of a cost-space path-planning algorithm together with some results deriving from the static analysis of cable-driven manipulators. Based on the so-called wrench-feasibility constraints applied to the cable tensions, as well as thrust constraints applied to the flying robots, we characterize the set of feasible configurations of the system. Besides, the expression of these constraints leads to a criterion to evaluate the quality of a configuration. This allows us to define a cost function over the configuration space, which we exploit to compute high-quality paths using T-RRT. As part of our approach, we also propose an aerial towed-cable system that we name *FlyCrane*. It consists of a platform attached to three flying robots using six fixed-length cables. We validate the proposed approach on two simulated 6-D quasi-static manipulation problems involving such a system, and we show the benefits of taking the cost function into account for such path-planning tasks.

Second, we propose a new variant of T-RRT called *Anytime Multi-T-RRT*, based on the combination of two extensions of T-RRT: the Multi-T-RRT (cf. Chapter 3) and the Anytime T-RRT (cf. Chapter 4). This algorithm is especially useful to solve ordering-and-pathfinding problems, i.e. to compute a high-quality path going through several waypoints that are not a priori ordered. Using the Anytime Multi-T-RRT, such problems can be solved from a purely geometrical perspective, without having to utilize a symbolic task planner. To demonstrate its capabilities, we apply the Anytime Multi-T-RRT to a concrete industrial inspection problem involving an aerial robot.

These two kinds of problems are typical of what the ARCAS¹ project addresses. One of the goals of this project is to develop robot systems for the assembly, inspection and maintenance of industrial installations difficult to access for humans.

¹<http://www.arcas-project.eu>

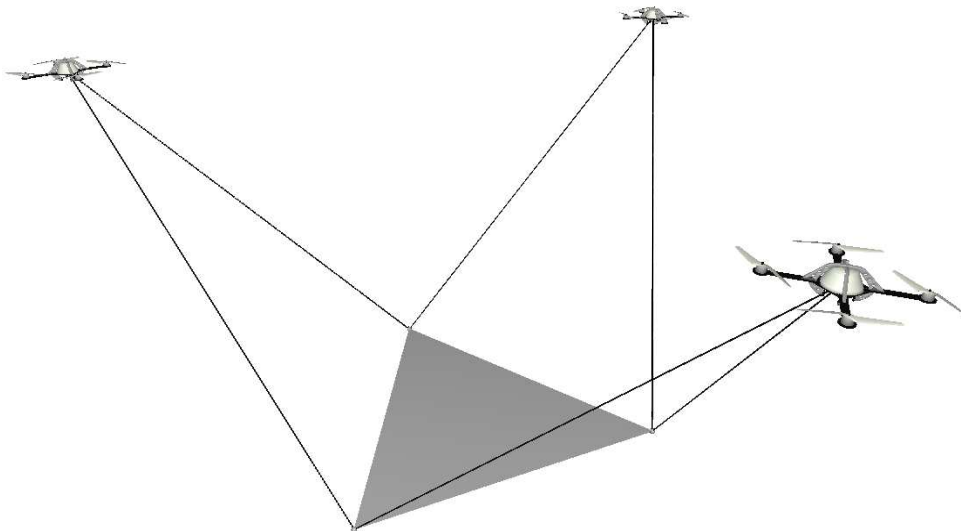


Figure 6.1: Octahedral version of the *FlyCrane* system.

6.1 6-D Manipulation with an Aerial Towed-Cable System

Aerial towed-cable systems have been used for decades, mainly as crane devices. They have proven to be useful in various contexts, such as supply delivery missions and rescue operations [14], as well as environmental monitoring and surveillance [159]. One such system has even been successful as a safe soft-landing device for a rover on the martian surface [147], for instance. In all these examples, the systems only required a certain accuracy in position, for example to execute simple trajectories [123, 146]. Little work has been done on trying to control a load both in position and in orientation. To the best of our knowledge, the only existing technique for 6-dimensional manipulation with an aerial towed-cable system requires a discrete set of load poses to be provided [62, 121]. Such a technique relies on solving the inverse kinematics problem and determining the static equilibrium for all given poses. Requiring a given set of platform poses may be too restrictive, though, especially in constrained workspaces: it may yield no result, while there may exist solutions for other intermediate poses.

In this section, we propose a new reliable path planning approach for 6-D quasi-static manipulation with aerial towed-cable systems. The method only requires a start and goal configurations as input, and provides a feasible path to achieve the manipulation task. In addition to being feasible, the generated manipulation path has high quality, meaning that all intermediate configurations fulfill adequate physical properties related to the forces applied to the aerial system and to cable tensions. This quality is measured by a formal cost function derived from the static analysis of the system, based on a formulation similar to that used for cable-driven manipulators [18, 21]. Then, as it can take this cost function into account, the T-RRT algorithm is applied to compute high-quality paths, i.e. feasible paths that do not approach dangerous/uncontrollable configurations of the system. It is worth noting that, to the best of our knowledge, this is the first time T-RRT is applied to aerial manipulation problems.

In addition to the methodology, this section presents an aerial towed-cable system that can perform 6-D manipulation tasks, and that we call the *FlyCrane*. This system consists of a moving platform attached to three flying robots by means of six fixed-length cables linked by pairs to each robot. The 6-D manipulation of the platform is performed by varying the relative positions of the flying robots. An octahedral version of this system is illustrated in Figure 6.1.

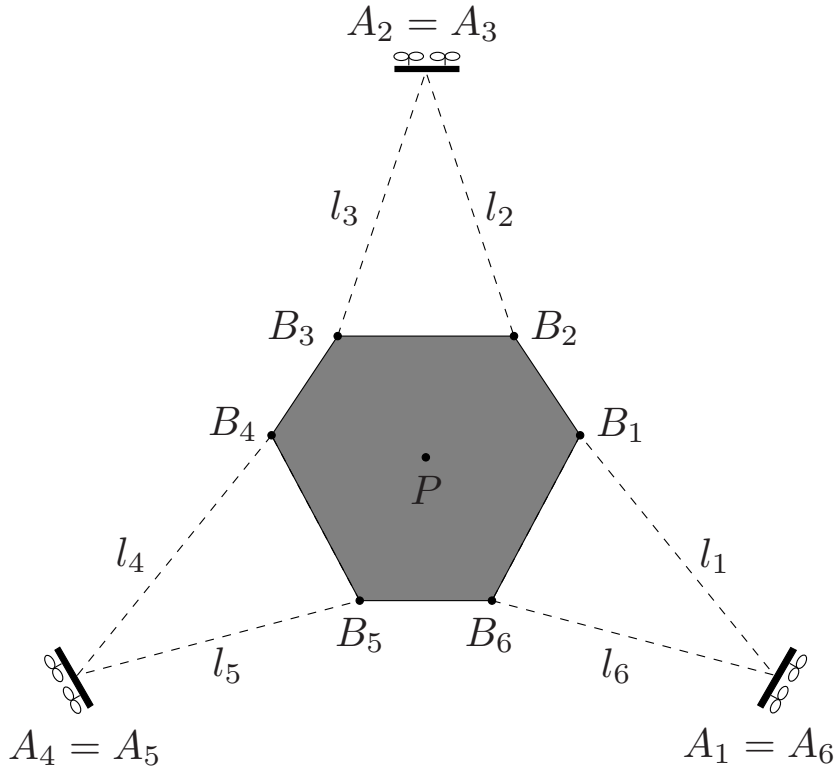


Figure 6.2: Geometric structure of a generic *FlyCrane* system.

6.1.1 Path Planning Approach

Towed-cable systems present important analogies with cable-driven manipulators. This allows the static analysis of towed-cable systems to be performed in a way similar to what is done for cable-driven manipulators [18]. However, while cable-driven manipulators have to adjust the lengths of their cables to reach a precise pose of the platform, towed-cable systems have fixed-length cables and are actuated by displacing their anchor points. Manipulating the six degrees of freedom of a load requires a minimum of seven cables, unless convenient forces reduce this number. In the case of crane systems, for instance, gravity acts as an implicit cable, and therefore six cables suffice for a full 6-D manipulation. Examples of such systems are the NIST robocrane [4] or more general cable-driven hexapods [18].

In the aerial towed-cable system we propose, and that we call the *FlyCrane*, the platform is also pulled by six cables, which, as illustrated in Fig. 6.2, are attached pairwise to three flying robots (instead of being individually attached to six flying robots). It is worth noting that three is the minimal number of flying robots required to properly operate this system, as less robots would not allow for the manipulation of the six degrees of freedom of the platform. Whenever the base points of the cables are also coupled (i.e. when $B_1 = B_2$, $B_3 = B_4$ and $B_5 = B_6$), we call this system the *octahedral FlyCrane*. Indeed, the resulting structure can be seen as an octahedron comprising the following eight triangles: the triangle formed by the base points of the cables, the triangle formed by the flying robots, and the six triangles corresponding to the pairs of adjacent cables. For a more detailed description of the *FlyCrane*, the interested reader is referred to [116].

Note that we assume, in this section, that motions are performed quasi-statically; we therefore ignore the dynamic analysis of this aerial system. Although it may appear as a strong simplification, this assumption is frequently made in fine-positioning situations, where slow motion is imperative. Nevertheless, dealing with dynamical aspects is an interesting direction of future research.

Even with six cables, the six degrees of freedom of the platform can be governed only in a subset of the configuration space of the system. Indeed, the pose of the platform is locally determined only when all cables are in tension. Therefore, it is important to prevent the cables from being slack or too tight. Besides, the flying robots must be able to counteract the forces exerted on them. These two conditions determine the feasibility of a configuration of the system. More precisely, to be feasible, a configuration must satisfy the following two types of constraints:

- *Wrench-feasibility* constraints: they guarantee that the system is able to statically counteract a set of wrenches applied on the platform while ensuring that the cable tensions always lie within a pre-defined, positive acceptance range; they are derived from the static analysis of cable-driven manipulators [18,21].
- *Thrust* constraints: they guarantee that the thrust of the flying robots can equilibrate the forces applied on them, namely the forces exerted by the cables and the force of gravity.

For a detailed and formal definition of these feasibility constraints, the interested reader is referred to [116].

The current aim of the *FlyCrane* system is the 6-D quasi-static manipulation of a load. The resolution of such manipulation problem can be seen as a path-planning query with the addition of the aforementioned feasibility constraints. Note that paths have to be found in the manifold formed by the configurations satisfying these feasibility constraints. Furthermore, an infinite number of feasible solution paths may exist for a given manipulation query. However, the desired manipulation motions should avoid solution paths that may approach the violation of the feasibility constraints. A way to discriminate the less appropriate paths is to define a criterion assessing their quality. A high-quality path should be a path whose intermediate configurations are attributed a low cost with respect to the physical properties of the system. With this in mind, we define a quality measure of the configurations of the system, given as a cost function $c : \mathcal{C} \rightarrow \mathbb{R}_+$.

A meaningful way to evaluate the cost of a configuration of the system is to derive it from the previous feasibility constraints. The idea is to define a cost function that tends toward infinity when a configuration approaches the limit of a feasibility constraint (i.e. when a cable tension approaches one of its limits, or when a robot approaches its maximum thrust) and that takes low positive values when a configuration is far from the non-feasible ones. Such cost function can be written as a combination of terms deriving from the equations that define the aforementioned feasibility constraints. More precisely, the cost of a configuration is defined as the sum of the inverses of the following terms:

- for each cable, we compute the differences between the current tension exerted on this cable and the bounds of its interval of admissible tensions
- for each flying robot, we compute the difference between the current thrust of this robot and its maximal thrust.

For a detailed and formal definition of this configuration-cost function, the interested reader is referred to [116].

Any sampling-based path planner, such as RRT, could be applied to compute collision-free paths satisfying the previous feasibility constraints and performing 6-D manipulation tasks with the *FlyCrane* system. However, RRT might produce low-quality paths (as shown in the next section) because it cannot take the cost function into account. Therefore, we base our path-planning strategy on using T-RRT.

Table 6.1: Evaluation of RRT and T-RRT on the *Puzzle* and *Rescue* problems. Average values over 100 runs are given for: the average cost $avgC$, the maximal cost $maxC$, the mechanical work MW , the integral of the cost IC , the running time t (in seconds), the number of nodes N in the tree, and the number of expansion attempts X .

		$avgC$	$maxC$	MW	IC	t (s)	N	X
<i>Puzzle</i>	RRT	113	1,170	1,200	3,000	30	2,700	16,000
	T-RRT	8	23	19	300	170	4,700	79,000
<i>Rescue</i>	RRT	10	58	55	810	130	1,400	190,000
	T-RRT	4	4	1	250	50	400	210,000

6.1.2 Test Cases

We evaluate the proposed approach on two 6-D quasi-static manipulation problems involving the octahedral *FlyCrane* system and an equilateral platform. The first example is a complex task (inspired by classical motion planning benchmarks) in which the *FlyCrane* has to get a 3-D puzzle piece through a hole, as illustrated by Fig. 6.3. The second example, presented in Fig. 6.4, simulates a more realistic situation in which the *FlyCrane* has to install a lightweight footbridge between two buildings to evacuate people during a rescue operation. These examples differ in terms of difficulty: the *Rescue* problem is the easiest one because it requires only a translation and two rotations about a single axis of the *FlyCrane* for a solution to be found; the *Puzzle* problem requires to simultaneously perform a translation and four rotations about two axes of the *FlyCrane*.

On both examples, we evaluate the performance of the RRT and T-RRT algorithms on the basis of their running time t (in seconds), the number of attempted expansions X , and the number of nodes N in the produced tree. To avoid generating trivially-non-feasible paths, RRT only accepts feasible configurations, i.e. collision-free configurations satisfying the aforementioned feasibility constraints. After performing a smoothing operation (based on the random shortcut method [69]) on the paths generated by RRT and T-RRT, we evaluate path quality by computing the average cost $avgC$, the maximal cost $maxC$, the mechanical work MW , and the integral of the cost IC (cf. Section 2.3.2). For all variables, we give values averaged over 100 runs. Results were obtained on an Intel Core i5 processor at 2.6 GHz with 8 GB of memory.

Table 6.1 contains the results of our evaluation of RRT and T-RRT on the *Puzzle* and *Rescue* problems. Unsurprisingly, it shows that T-RRT provides higher-quality paths than RRT on both examples: on the *Puzzle* problem, all cost statistics are between 10 and 100 times lower for paths generated by T-RRT; on the *Rescue* problem, they are between 3 and 50 times lower. Since it generally requires more expansion attempts to find configurations with acceptable cost, T-RRT is often slower than RRT. This is what happens on the *Puzzle* problem where RRT runs faster than T-RRT (30 s vs 170 s) mainly because it performs less expansions (16,000 vs 79,000). On the contrary, it is worth noting that T-RRT runs faster than RRT on the *Rescue* problem (50 s vs 130 s). This is a consequence of the beneficial filtering and biasing effects of the transition test of T-RRT. Indeed, we observe that, even though T-RRT and RRT perform a similar number of expansions (210,000 vs 190,000), T-RRT produces less nodes than RRT (400 vs 1,400).

It is interesting to analyze what makes path quality differ when planning paths with RRT or T-RRT. For that, we compute the tensions exerted on each cable and the forces exerted on each quadrotor, along the paths produced by RRT and T-RRT, after dividing every path into 100 steps corresponding to intermediate configurations of the system. Then, for each path-step, we compute the minimal and maximal tensions (over all cables) and forces (over all quadrotors) over the 100 paths produced by RRT and over the 100 paths produced by T-

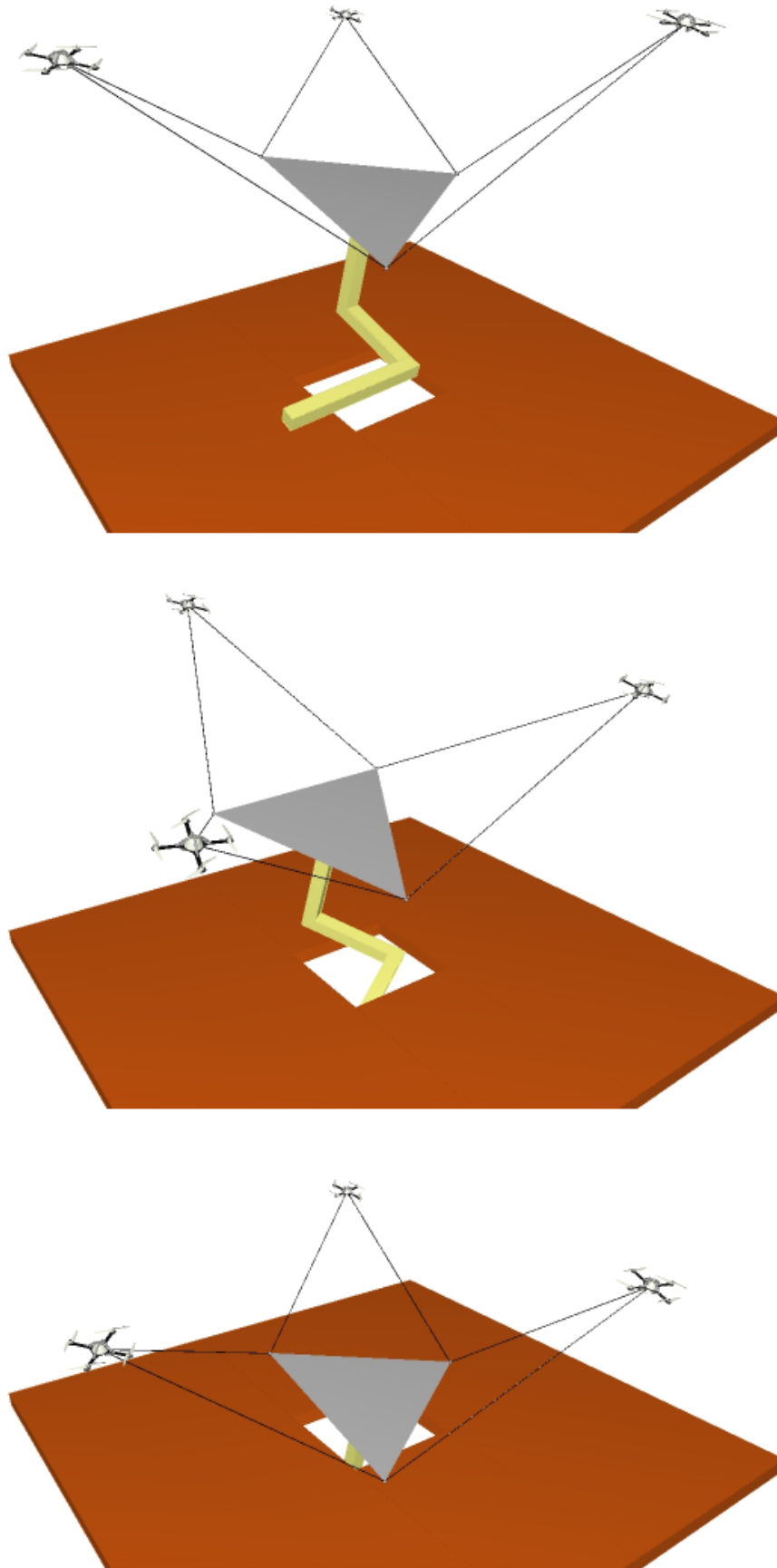


Figure 6.3: The *Puzzle* problem: the *FlyCrane* has to get a 3-D puzzle piece through a hole.

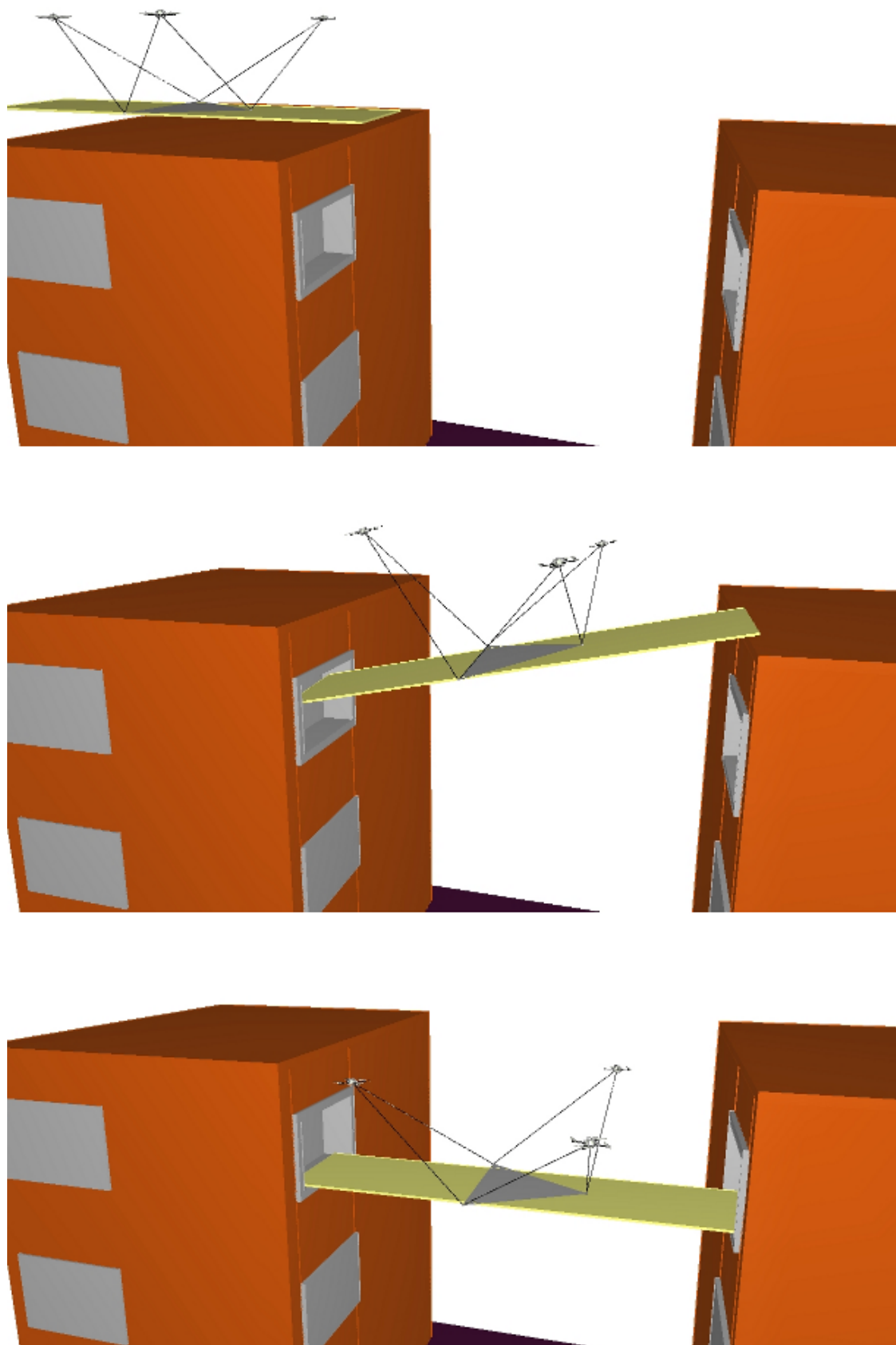


Figure 6.4: The *Rescue* problem: the *FlyCrane* has to install a lightweight footbridge between two buildings for a rescue operation.

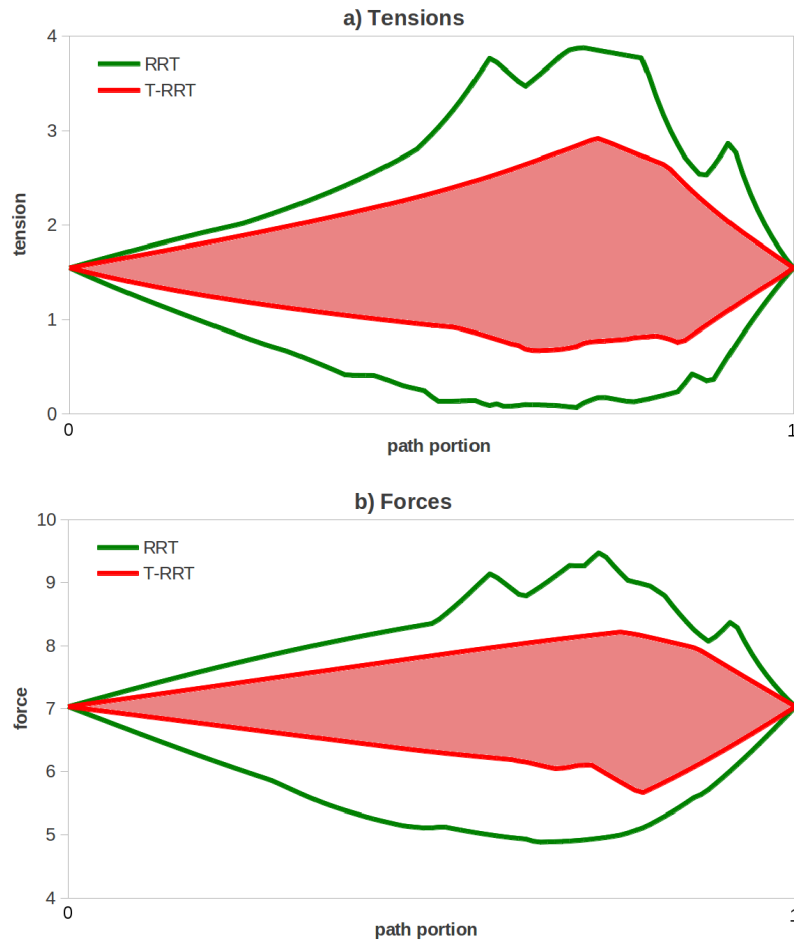


Figure 6.5: Profiles of a) the tension range and b) the force range, observed over 100 paths produced by RRT and T-RRT on the *Rescue* problem. The filled areas between the red curves represent the ranges for T-RRT; the areas between the green curves represent the ranges for RRT. Plots obtained on the *Puzzle* problem are similar.

RRT. Therefore, for each path-step, we obtain the tension ranges and the force ranges yielded by RRT and T-RRT respectively. Fig. 6.5 presents the profiles of the tension range and of the force range, respectively, on the *Rescue* problem. Similar plots have been obtained on the *Puzzle* problem. We can see that using T-RRT leads to smaller tension and force ranges than using RRT. More importantly, we observe that RRT produces paths along which a tension or a force can be dangerously close to a bound of its validity interval. For example, Fig. 6.5.a shows that, along some path, at least one tension comes close to zero, meaning that at least one cable almost goes slack. Similarly, on the *Puzzle* problem, one force comes close to the maximal thrust value (not shown here). Therefore, we argue that using a cost-space path planner, such as T-RRT, allows planning safer paths for the *FlyCrane* system than using a simple path planner, such as RRT.

6.1.3 Conclusion

We have presented an approach for the 6-D quasi-static manipulation of a load with an aerial towed-cable system. The main contribution of the approach lies in the combination of results deriving from the static analysis of cable-driven manipulators with the application of a cost-space path-planning algorithm to solve manipulation queries. The link underlying this combination is the definition of a quality measure for the configurations of the system.

First, this quality measure is based on wrench-feasibility constraints similar to those applied to cable-driven manipulators, and on additional thrust constraints. It allows us to 1) discriminate non-feasible from feasible configurations, and 2) favor configurations that are far from violating these constraints, by attributing them a low cost. Second, this quality measure leads to the definition of a configuration-cost function, thus allowing for the use of a cost-space path planner, such as T-RRT. As a result, rather than simply computing collision-free paths, the proposed approach produces high-quality paths, with respect to the constraints imposed on the system.

As part of our approach, we have additionally proposed an aerial towed-cable system that we have named the *FlyCrane*. This system consists of a platform attached to three flying robots by means of three pairs of fixed-length cables. We have evaluated the approach, in simulation, on two 6-D manipulation problems involving an octahedral version of the *FlyCrane* system. Results show that the proposed path planning approach is suitable to solve 6-D quasi-static manipulation tasks. They have also confirmed that RRT may produce paths that occasionally approach dangerous situations, while T-RRT produces safer paths. Furthermore, we have observed that, because of the complexity of the constraints involved, T-RRT can find a solution path faster than RRT on some problems, which is usually not the case with simpler systems incorporating simpler cost functions. To summarize, this work clearly illustrates the interest of cost-space path planning over feasible path planning when dealing with complex robotic systems.

The proposed approach allows for extensions in several ways. In particular, we expect to extend the method to consider positioning errors for the flying robots, which could be due to external force perturbations and to errors in the localization methods. Additionally, an interesting and challenging extension to this work is the introduction of dynamics in the motion of the load and of the flying robots, as they play an important role in the overall manipulation of the system.

In this section, we have applied the proposed approach in simulated environments. As part of our future work, we plan to implement it in a real aerial towed-cable system. This will serve as a testbed to validate the method and its further extensions, providing relevant feedback on the real limitations of the approach and of the system. In real-life situations, this approach could be helpful in various contexts. As illustrated by the simulated *Rescue* problem, one possible application is the construction of platforms for the evacuation of people in rescue operations. Another application could be the installation of platforms in uneven terrains for the landing of manned or unmanned aircrafts. More generally, it could be useful for the assembly of structures in places difficult to access for humans.

Finally, an interesting prospect is to combine the work presented in this section (which focuses on the definition of an application-specific cost function) with the algorithmic results presented in previous chapters. Indeed, the approach involving the *FlyCrane* could be enriched by using some of the more sophisticated variants of T-RRT we have developed, such as the Anytime T-RRT (cf. Chapter 4) or the Multi-T-RRT (cf. Chapter 3). This is particularly true for the Anytime T-RRT, as it would allow us to produce the optimal path to perform a given manipulation task.

6.2 Ordering-and-Pathfinding Problems

In this section, we aim at going beyond simple “init-to-goal” path-planning problems. Our objective is to generate high-quality paths going through a set of unordered waypoints in a cost space. Note that “multi-waypoints” problems have already been studied in the context of optimal path planning, with pre-ordered waypoints [89, 113, 119, 161, 165] or unordered waypoints [8, 43, 138, 141, 150]. However, most approaches focus on minimizing path length or path duration [8, 43, 89, 113, 119, 138, 141, 150], and very few consider more sophisticated quality criteria [161, 165]. To the best of our knowledge, this work is the first attempt to address “multi-waypoints” problems with unordered waypoints in the context of cost-space path planning, i.e. using a cost-based quality criterion.

Because of their dual nature, instead of referring to these path planning problems as being “multi-waypoints” problems, we call them “ordering-and-pathfinding” problems. Indeed, they involve two separate aspects: a low-level path planning problem that consists of connecting pairs of waypoints via high-quality paths (i.e. low-cost paths), and a high-level ordering problem that consists of finding an efficient way to visit all the waypoints (which is a simple kind of task planning problem).

Hybrid approaches to solve task-and-path planning problems are often based on decoupling the two aspects: a symbolic task planner computes a high-level plan (possibly based on geometrical data) that is refined by a geometric path planner computing precise low-level paths [24, 88, 96]. Other approaches aim at further interleaving the task planning and path planning levels, e.g. by using temporal constraints [120, 133]. In some cases, when tasks are simple enough, the overall problem possesses a purely geometrical formulation, and no symbolic task planner is needed [144].

In this work, we also follow a purely geometrical approach and we apply it to a cost space: the geometric path planner yields high-quality high-level solutions based on the costs of the low-level paths it computes between waypoints. To achieve that, we combine two extensions of T-RRT presented in previous chapters, namely the Multi-T-RRT (see Section 3.3) and the Anytime T-RRT (see Chapter 4), which produces the *Anytime Multi-T-RRT*. The objective is to quickly obtain a high-quality solution path going through all the waypoints, and then continually improve it in an anytime fashion. In this way, the current solution-path can converge toward the optimal path.

6.2.1 The Anytime Multi-T-RRT Algorithm

The pseudo-code of the Anytime Multi-T-RRT is presented in Algorithm 19. It consists of first running the Multi-T-RRT until all trees rooted at the n waypoints are merged into a single tree (lines 1–14). If this succeeds, meaning that n has been decreased to 1 (line 15), there exists in the tree \mathcal{T} a path going through all the waypoints. Then, if some running time is still available, the Anytime T-RRT is used to improve the current solution path by carrying on the exploration and adding cycles to the graph \mathcal{G} (lines 16–24).

The role of the `extractPathFromGraph` procedure is to extract from \mathcal{G} the path having minimal cost and going through all the waypoints. No symbolic task planner is required if we consider that this is simply an instance of the Traveling Salesman Problem (TSP) involving the complete graph² whose nodes are the waypoints. The distance associated with an edge of this graph can be estimated as the cost of the lowest-cost path between two waypoints in \mathcal{G} . When only few waypoints are defined, the TSP is solved by an exhaustive search among all sequences. When more waypoints are involved, the Nearest-Neighbor or Multi-Fragment heuristics are used [10].

²Note that this can be a graph or a digraph, depending on the criterion used to assess path quality. If we use a symmetric quality criterion, such as IC, we obtain a graph. If we use a quality criterion that is not symmetric, such as MW, we obtain a digraph.

Algorithm 19: Anytime Multi-T-RRT

```

input : the optimal path planning problem  $(\mathcal{C}, \{q_{\text{init}}^k \mid k = 1..n\}, c, c_p)$ 
         where  $q_{\text{init}}^k, k = 1..n$  are a set of unordered waypoints
output: the path  $\mathcal{P}$  going through all the waypoints
1 for  $k = 1..n$  do
2    $\mathcal{T}_k \leftarrow \text{initTree}(q_{\text{init}}^k)$ 
3 while not  $\text{stoppingCriteria}(\{\mathcal{T}_k \mid k = 1..n\})$  do
4    $\mathcal{T}' \leftarrow \text{chooseNextTreeToExpand}()$ 
5    $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
6    $q'_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}', q_{\text{rand}})$ 
7    $q_{\text{new}} \leftarrow \text{extend}(q'_{\text{near}}, q_{\text{rand}})$ 
8   if  $q_{\text{new}} \neq \text{null}$  and  $\text{transitionTest}(\mathcal{T}', c(q'_{\text{near}}), c(q_{\text{new}}))$  then
9      $\text{addNewNode}(\mathcal{T}', q_{\text{new}})$ 
10     $\text{addNewEdge}(\mathcal{T}', q'_{\text{near}}, q_{\text{new}})$ 
11     $(\mathcal{T}'', q''_{\text{near}}) \leftarrow \text{findNearestTree}(q_{\text{new}})$ 
12     $\mathcal{T} \leftarrow \text{attemptLink}(\mathcal{T}', q_{\text{new}}, \mathcal{T}'', q''_{\text{near}}, n)$ 
13 if  $n = 1$  then
14    $\mathcal{G} \leftarrow \text{initGraph}(\mathcal{T})$ 
15   while not  $\text{stoppingCriteria}(\mathcal{G})$  do
16      $q_{\text{rand}} \leftarrow \text{sampleRandomConfiguration}(\mathcal{C})$ 
17      $q_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{G}, q_{\text{rand}})$ 
18      $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}, q_{\text{rand}})$ 
19     if  $q_{\text{new}} \neq \text{null}$  and  $\text{transitionTest}(\mathcal{G}, c(q_{\text{near}}), c(q_{\text{new}}))$  then
20        $\text{addNewNode}(\mathcal{G}, q_{\text{new}})$ 
21        $\text{addNewEdge}(\mathcal{G}, q_{\text{near}}, q_{\text{new}})$ 
22        $\text{addUsefulCycles}(\mathcal{G}, q_{\text{new}}, c_p)$ 
23    $\mathcal{P} \leftarrow \text{extractPathFromGraph}(\mathcal{G}, \{q_{\text{init}}^k \mid k = 1..n\}, c_p)$ 
24   return  $\mathcal{P}$ 
25 else
26   return  $\text{null}$ 

```

As an example, Fig. 6.6 shows the interest of using the Anytime Multi-T-RRT on the *Stones-large* problem, where ten waypoints are defined without explicit order. The upper part of Fig. 6.6 shows a path that is representative of what is generated by the Multi-T-RRT, in about 0.3 s (on average over 100 runs). It features many portions along which the disk would have to go forward and then backward, which is a very inefficient way to visit the waypoints. The lower part of Fig. 6.6 shows a path that is representative of what the Anytime Multi-T-RRT produces for a running time of 5 s (as observed over 100 runs). Its cost (based on IC) is about half the cost of the first path. From the labels, we can also see that this path provides a much more efficient way to visit all the waypoints than the first path.

6.2.2 Application to Industrial Inspection

Based on this ordering-and-pathfinding approach, we present an industrial inspection problem involving an aerial robot in a dense environment, as illustrated by Fig. 6.7. This example is a multi-waypoints variant of a similar example involved in the experimental evaluations of Chapters 3 and 4. The version used in previous chapters is a simpler “init-to-goal” path-planning problem whose main interest is to feature a large-scale workspace on which we can

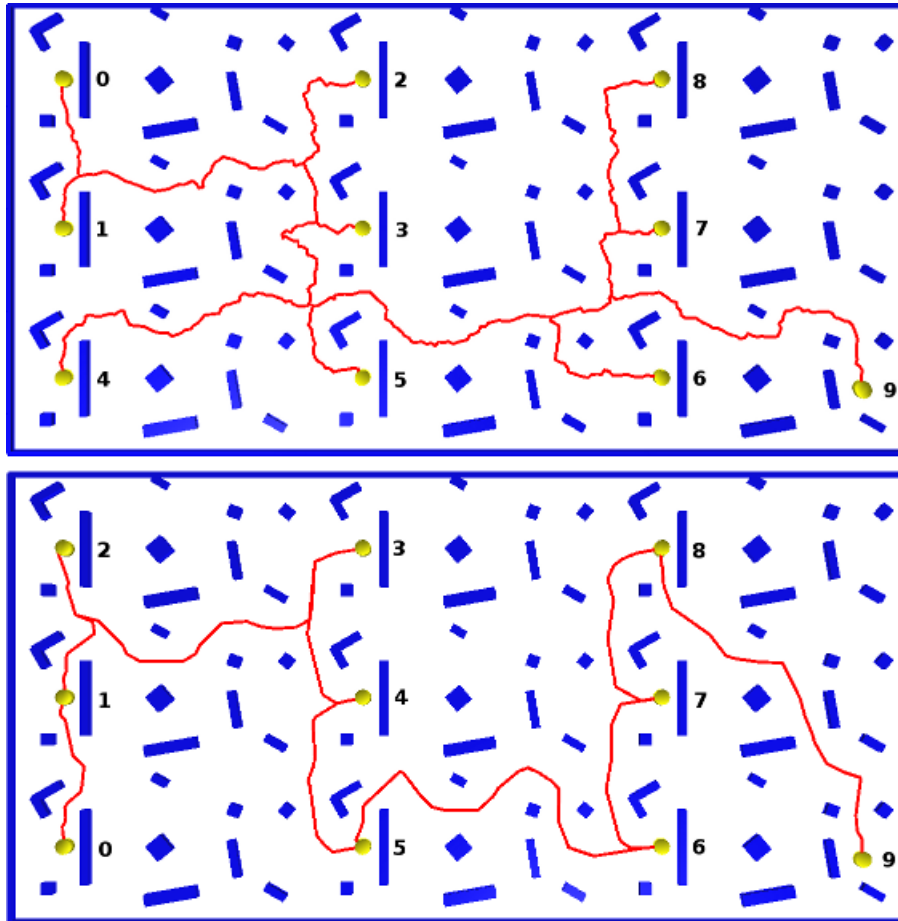


Figure 6.6: The *Stones-large* problem, featuring ten waypoints (that are not ordered a priori). The cost is the inverse of the distance between the 2-DoF yellow disk and the blue rectangular-shaped obstacles. Top: path generated by the Multi-T-RRT in 0.3 s. Bottom: path produced by the Anytime Multi-T-RRT, after a running time of 5 s. The labels show the order in which waypoints are visited.

evaluate the scalability of path-planning algorithms. Here, the size of the workspace allows us to define a relevant and challenging “multi-waypoints” path-planning problem.

In this example, a quadrotor is used to inspect an oil platform, going through eight waypoints defined a priori without explicit order (cf. Fig. 6.7). The quadrotor is modeled as a 3-DoF sphere (i.e. a free-flying sphere) representing the security zone around it. For safety reasons, it has to move in this environment trying to maximize clearance. Therefore, the cost of a configuration is the inverse of the distance between the quadrotor and the obstacles. Assuming that the motions of the quadrotor are performed quasi-statically, we restrict the problem to planning in position (thus ignoring controllability issues).

Even though this example features a large-scale workspace, the Anytime Multi-T-RRT can quickly provide a high-quality solution path going through all the waypoints in an efficient manner. For instance, the path featured in Fig. 6.7 has been computed in 50 s. Its cost (based on IC) is about half the cost of the first solution path generated by the Multi-T-RRT (and found in about 5 s). Contrary to existing approaches, that could allow computing the shortest path going through all the waypoints, using the Anytime Multi-T-RRT enables us to obtain a path along which clearance is maximized, while additionally optimizing the way all the waypoints are visited. Therefore, the solution paths produced by the Anytime Multi-T-RRT maintain a good balance between safety and efficiency.

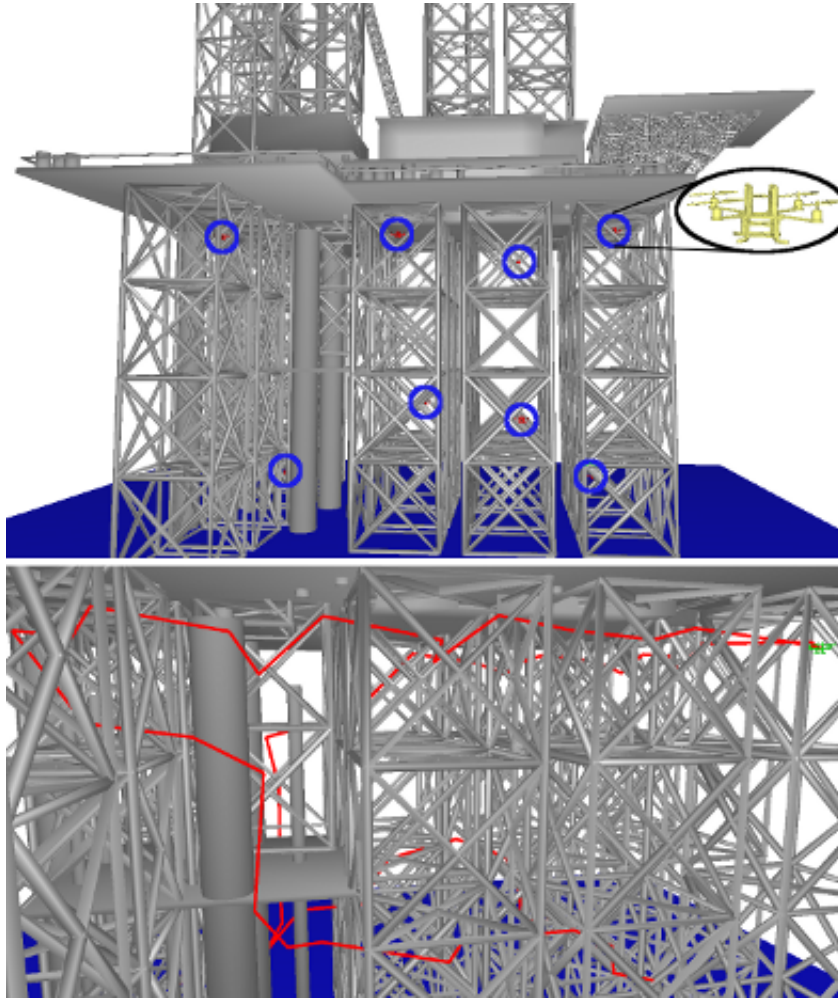


Figure 6.7: Top: eight waypoints (shown in red and circled in blue) defined for a quadrotor (whose close-up is shown in yellow) inspecting an oil platform. Bottom: example of a path efficiently visiting these waypoints, and produced in 50 s by the Anytime Multi-T-RRT.

6.2.3 Conclusion

We have developed a variant of T-RRT called *Anytime Multi-T-RRT*, and based on the combination of two extensions of T-RRT presented in previous chapters: the Multi-T-RRT and the Anytime T-RRT. The Multi-T-RRT can be used to compute a path going through a set of waypoints in a cost space. The Anytime T-RRT allows improving the current solution path and converging toward the optimal path.

In this section, we have focused on path-planning problems where the order of the waypoints is not defined a priori. Such problems, that we refer to as ordering-and-pathfinding problems, involve both task and path planning aspects. Using the Anytime Multi-T-RRT in this context allows us to find a way to visit the waypoints following a high-quality “global” path computed (without the use of a symbolic task planner) based on the costs of the “local” paths connecting pairs of waypoints. We have demonstrated the interest of this approach on a simulated yet realistic industrial inspection problem featuring a large-scale workspace and an aerial robot.

In the next chapter, we show another application for the Anytime Multi-T-RRT, in the field of structural biology. This domain is particularly challenging and could push our algorithm to its limits. Indeed, instead of growing a few trees over the search space, we might want to grow hundreds or even thousands of trees simultaneously.

Chapter 7

Application to Structural Biology Problems

In this chapter, we present two applications of sampling-based algorithms, within the field of computational structural biology. The main challenge in this field is that all problems are inherently difficult because of their high-dimensionality, even when only small molecules are considered. Many interesting research questions can be addressed using robotics-inspired algorithms because they allow for an efficient exploration of the conformational space of a molecule or even a molecular complex. Here, we focus on two different themes: the exploration of the energy landscapes of small peptides, and the simulation of the unbinding process of a protein-ligand complex.

First, we address the issue of obtaining a full characterization of energy landscapes of small yet highly-flexible peptides. For that, we suggest to combine two complementary sampling-based methods: the Basin Hopping and the (Anytime) Multi-T-RRT algorithms. We propose a simplified version of the classical Basin Hopping algorithm, that can quickly reveal the meta-stable structural states of a peptide and the corresponding low-energy basins in the landscape. Then, we use variants of the T-RRT algorithm to quickly determine transition state and transition path ensembles, as well as transition probabilities between these meta-stable states. More precisely, we propose an Anytime Multi-T-RRT tailored to structural biology problems, based on the Multi-T-RRT (cf. Chapter 3) and on the Anytime T-RRT (cf. Chapter 4). We demonstrate the approach combining Basin Hopping and T-RRT on the terminally-blocked alanine.

Second, we tackle the problem of simulating protein-ligand interactions taking place far away from the active site of the protein, during ligand binding or release. As a first step toward what we want to achieve, the approach we present here is purely geometric. It is based on a mechanistic representation of the molecular system, considering partial flexibility, and on the application of a variant of RRT, called Manhattan-like RRT (ML-RRT), to explore the conformational space. This means that no molecular energy is computed, and that motions are validated only on the basis of collision avoidance. Such a purely geometric approach, together with the efficiency of the exploration algorithm, enables the simulation of ligand unbinding within very short computing time. Achieving low runtime is a constraint we had imposed on ourselves, with the objective of developing an efficient web server. This web tool yields ligand unbinding pathways that, as a first approximation, can provide useful information about protein-ligand interactions. We show the interest of this computational tool on the hexameric insulin-phenol complex.

All the algorithms presented in this chapter have been developed in a computer software called MoMA (for Molecular Motion Algorithms), which implements a collection of robotics-inspired algorithms for the simulation of molecular motions.

7.1 Exploration of the Energy Landscape of a Small Peptide

Global thermodynamic and kinetic properties of molecules can be extracted from an analysis of their conformational energy landscapes [155]. In particular, obtaining an accurate representation of a molecule’s energy landscape is a significant first step to conducting detailed structure-function studies for bio-molecules of central importance in the cell, such as proteins and peptides [143].

In this work, we focus on small peptides. Despite their modest size, they represent in many ways a more challenging setting than larger proteins. Peptides exhibit high structural flexibility, which enables them to recognize different molecular partners in the cell, and thus to modulate their biological function [128]. Contrary to proteins, that are often characterized by a unique native state and a funnel-shaped energy landscape, peptides are characterized by several meta-stable structural states; their energy landscape may contain a multitude of competitive low-energy basins.

The existence of multiple local minima in a molecule’s energy landscape makes it particularly challenging to map this landscape and reconstruct all the functionally-important regions in it. Experimental methods, such as X-ray crystallography or nuclear magnetic resonance (NMR), cannot reveal such maps, as they can uncover only few structures at best [7]. It is therefore the task of computational techniques to obtain detailed representations of energy landscapes.

Currently, only sample-based representations of the energy landscape can be afforded. Even for small peptides, the space of possible conformations is vast, and the degrees of freedom needed to represent a conformation are numerous. The high dimensionality of the space is accompanied by a complex (non-linear, non-convex) expression for the conformational energy, which is the result of competing local and non-local inter-atomic interactions. Probing this landscape is thus very computationally-costly. Currently, only stochastic optimization techniques provide the right balance between accuracy and computational efficiency [143,155].

Obtaining a representation of an energy landscape can be divided into two sub-problems: (1) determining meta-stable structural states (i.e. local energy minima); (2) computing transition paths between the identified states. Both can be addressed by achieving a dense yet efficient sampling of the conformational space.

In this section, we propose to combine two sampling-based techniques to obtain a full characterization of energy landscapes of small yet highly-flexible peptides. First, we utilize our own variant of the Basin Hopping algorithm [156] to sample local minima in a peptide’s energy landscape, and characterize low-energy basins in this landscape. Local minima are then organized via density-based clustering to reveal meta-stable structural states. Second, we use the Multi-T-RRT and the Anytime Multi-T-RRT to map out the connectivity between these states, thus completing the reconstruction of the energy landscape. More precisely, we compute transition path and transition state ensembles, as well as transition probabilities between meta-stable structural states. To illustrate this, and as a proof-of-concept, we present results obtained on the terminally-blocked alanine, which is a frequent benchmark for studies in theoretical physical chemistry.

7.1.1 Methods

This work is motivated by recent studies showing that robotics-inspired sampling-based algorithms can be a good basis for efficient conformational sampling and exploration in structural biology [3, 71]. Among such algorithms, T-RRT has been involved in previous work, in a biology context [82]. In this section, we utilize two variants of T-RRT, called Multi-T-RRT and Anytime Multi-T-RRT. These algorithms are combined with an in-house variant of the Basin Hopping algorithm involving a simplified minimization process. Together they are used to obtain a complete representation of the energy landscape of a small peptide.

Algorithm 20: Anytime Multi-T-RRT

```

input : the conformational space  $\mathcal{C}$ 
         the energy function  $E : \mathcal{C} \rightarrow \mathbb{R}$ 
         the initial conformations  $q_{\text{init}}^k$ ,  $k = 1..n$ 
         the extension step-size  $\delta$ 

output: the graph  $\mathcal{G}$ 

1 for  $k = 1..n$  do
2    $\mathcal{T}_k \leftarrow \text{initTree}(q_{\text{init}}^k)$ 
3 while not stoppingCriteria( $\{\mathcal{T}_k \mid k = 1..n\}$ ) do
4    $\mathcal{T}' \leftarrow \text{chooseNextTreeToExpand}()$ 
5    $q_{\text{rand}} \leftarrow \text{sampleRandomConformation}(\mathcal{C})$ 
6    $q'_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{T}', q_{\text{rand}})$ 
7    $q_{\text{new}} \leftarrow \text{extend}(q'_{\text{near}}, q_{\text{rand}}, \delta)$ 
8   if  $q_{\text{new}} \neq \text{null}$  and transitionTest( $\mathcal{T}'$ ,  $E(q'_{\text{near}})$ ,  $E(q_{\text{new}})$ ) then
9     addNewNode( $\mathcal{T}'$ ,  $q_{\text{new}}$ )
10    addNewEdge( $\mathcal{T}'$ ,  $q'_{\text{near}}$ ,  $q_{\text{new}}$ )
11     $(\mathcal{T}'', q''_{\text{near}}) \leftarrow \text{findNearestTree}(q_{\text{new}})$ 
12    if distance( $q_{\text{new}}$ ,  $q''_{\text{near}}$ )  $\leq \delta$  then
13       $\mathcal{T} \leftarrow \text{linkAndMerge}(\mathcal{T}', q_{\text{new}}, \mathcal{T}'', q''_{\text{near}})$ 
14       $n \leftarrow n - 1$ 
15 if  $n = 1$  then
16    $\mathcal{G} \leftarrow \text{initGraph}(\mathcal{T})$ 
17   while not stoppingCriteria( $\mathcal{G}$ ) do
18      $q_{\text{rand}} \leftarrow \text{sampleRandomConformation}(\mathcal{C})$ 
19      $q_{\text{near}} \leftarrow \text{findNearestNeighbor}(\mathcal{G}, q_{\text{rand}})$ 
20      $q_{\text{new}} \leftarrow \text{extend}(q_{\text{near}}, q_{\text{rand}}, \delta)$ 
21     if  $q_{\text{new}} \neq \text{null}$  and transitionTest( $\mathcal{G}$ ,  $E(q_{\text{near}})$ ,  $E(q_{\text{new}})$ ) then
22       addNewNode( $\mathcal{G}$ ,  $q_{\text{new}}$ )
23       addNewEdge( $\mathcal{G}$ ,  $q_{\text{near}}$ ,  $q_{\text{new}}$ )
24       for  $q_{\text{can}} \in \mathcal{G} \setminus \{q_{\text{new}}\}$  do
25         if distance( $q_{\text{new}}$ ,  $q_{\text{can}}$ )  $\leq \delta$  and noEdgeBetween( $q_{\text{new}}$ ,  $q_{\text{can}}$ ) then
26           addNewEdge( $\mathcal{G}$ ,  $q_{\text{new}}$ ,  $q_{\text{can}}$ )
27   return  $\mathcal{G}$ 
28 else
29   return null

```

Anytime Multi-T-RRT

The variants of T-RRT we use in the structural biology context are slightly different from those used in the robotics context. Moreover, the terminology differs slightly. For instance, instead of talking about configuration space, we talk about *conformational space*, i.e. the space of all the conformations of a molecule. The cost function defined on this space associates to each conformation its molecular energy. In this context, we propose an *Anytime Multi-T-RRT*, based on the Multi-T-RRT (cf. Section 3.3) and on the Anytime T-RRT (cf. Section 4.1). The pseudo-code of the Anytime Multi-T-RRT is presented in Algorithm 20. The main differences with its robotics counterpart are the way connections are attempted between trees (lines 12–14) and the way useful cycles are created (lines 24–26). As we focus on developing an Extend

Algorithm 21: transitionTest (\mathcal{G} , E_i , E_j)

input : the energy threshold E_{\max}
the current temperature T
the temperature increase rate T_{rate}
the Boltzmann constant K

output: *true* if the transition is accepted, *false* otherwise

```

1 if  $E_j > E_{\max}$  then
2   | return False
3 else if  $E_j \leq E_i$  then
4   | return True
5 else if  $e^{-(E_j - E_i) / (K \cdot T)} > 0.5$  then
6   |  $T \leftarrow T / 2^{(E_j - E_i) / (0.1 \cdot \text{costRange}(\mathcal{G}))}$ 
7   | return True
8 else
9   |  $T \leftarrow T \cdot 2^{T_{\text{rate}}}$ 
10  | return False

```

version of the algorithm, for now, both operations are based on using the extension-step size δ . During the initial construction of the trees, if the distance between a new node and its closest neighbor in another tree is no more than δ , the two trees involved are linked and merged into one. In the anytime part of the algorithm, if the distance between a new node and another node in the graph is no more than δ , an edge is added between these two nodes (if they are not already linked by an edge), thus creating a cycle. Finally, in the biology context, the Metropolis-like expression in the transition test of T-RRT includes the Boltzmann constant (as shown in Algorithm 21, line 5).

In this section, we use the mechanical work to assess path quality. Other criteria could be considered, such as the minimum resistance (linked to the MaxFlux algorithm [76, 77]) or the maximum flux [163]. However, finding which criterion is the most relevant in this context is out of the scope of this thesis. Furthermore, it has been shown that, in the case of the terminally-blocked alanine, transition paths produced using different criteria are often similar [163].

Basin Hopping

The Basin Hopping (BH) algorithm is a popular method for sampling local minima of an energy landscape. It was originally introduced to obtain the Lennard-Jones minima of small atom clusters [156]. Recently, BH has gained new attention to predict protein structure [126], and to find intermediate structures of chemical reactions [98]. The method consists of repeatedly applying a structural perturbation followed by an energy minimization, which yields a trajectory of minima. The result is a (discrete) coarse-grained representation of the energy landscape that can be seen as a collection of interpenetrating staircases.

Our variant of BH differs from the classical one in that it does not involve local, gradient-based minimizations, but relies on simple Monte-Carlo-based (MC-based) minimizations. Our implementation of BH (whose pseudo-code is presented in Algorithm 22) follows a random restart procedure performing several rounds, each one starting from a conformation randomly sampled in the search space. Every round builds a trajectory of minima by performing a succession of structural perturbations followed by MC-based minimizations. Every MC-based minimization starts from a conformation obtained by performing a large-amplitude perturbation of the minimum reached at the previous step, or from the random sample,

Algorithm 22: Basin Hopping

```

input : the conformational space  $\mathcal{C}$ 
         the number of rounds  $nbRounds$ 
         the number of Monte-Carlo-based minimizations  $nbMC$ 
output: the list of trajectories of minima  $\mathcal{L}$ 
1  $\mathcal{L} \leftarrow \phi$ 
2 for  $r = 1..nbRounds$  do
3    $\mathcal{T} \leftarrow \phi$ 
4    $q \leftarrow \text{sampleRandomConformation}(\mathcal{C})$ 
5   for  $m = 1..nbMC$  do
6     if  $m > 1$  then
7        $q \leftarrow \text{doLargeAmplitudePerturbation}(q)$ 
8        $q \leftarrow \text{doMonteCarloBasedMinimization}(q)$ 
9        $\text{addMinimum}(\mathcal{T}, q)$ 
10   $\text{addTrajectory}(\mathcal{L}, \mathcal{T})$ 
11 return  $\mathcal{L}$ 

```

in the first step. An MC-based minimization is an iterative succession of small-amplitude perturbations. The search goes on, accepting new conformations if the energy decreases or the Metropolis criterion is satisfied, until a given number of consecutive rejections is reached. Every MC-based minimization produces a low-energy conformation that we call a “minimum” in a minor abuse of language. Every round produces what we call a milestone: the minimum (along the trajectory) having the lowest energy.

All the milestones (or the minima) produced by BH have to be grouped to provide a comprehensible list of meta-stable structural states. This clustering can be done in several ways. In this work, we perform a density-based clustering requiring the user to define the distance threshold between two clusters [97]. Distance computations are based on Root-Mean-Square Deviation (RMSD) and involve dihedral angles or atom coordinates.

Force Field

To compute conformational energy values, we use an in-house implementation of the AMBER parm96 force-field with an implicit representation of the solvent using the Generalized Born approximation. All results are obtained on an Intel Core i5 processor at 2.6 GHz with 8 GB of memory.

7.1.2 Results and Discussion

Terminally-Blocked Alanine

As a proof of concept, we use our methods to explore the potential energy landscape of the terminally-blocked alanine. This molecule is an alanine residue acetylated in its N-terminus and methylamidated in its C-terminus: Ace-Ala-Nme (cf. Fig. 7.1). Despite its small size, it is a common test-model in theoretical physical chemistry studies because of its complex energy landscape characterized by several local minima connected by multiple pathways involving various transition states [19, 32, 66, 125, 154].

The exploration of the conformational space is performed using an internal-coordinate representation of the terminally-blocked alanine, assuming constant bond lengths and bond angles. The conformational parameters are the dihedral angles $\{\phi, \psi, \chi\}$ of the Ala residue, χ of the Ace capping, χ of the Nme capping, ω of the Ace-Ala peptide bond, and ω of the

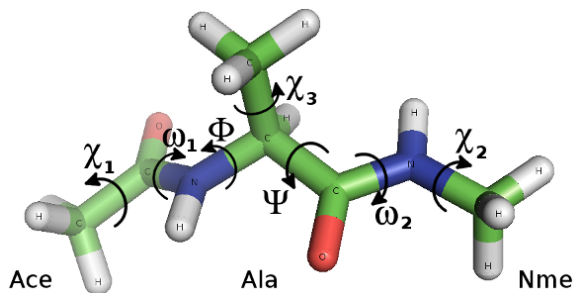


Figure 7.1: Internal-coordinate representation of the terminally-blocked alanine.

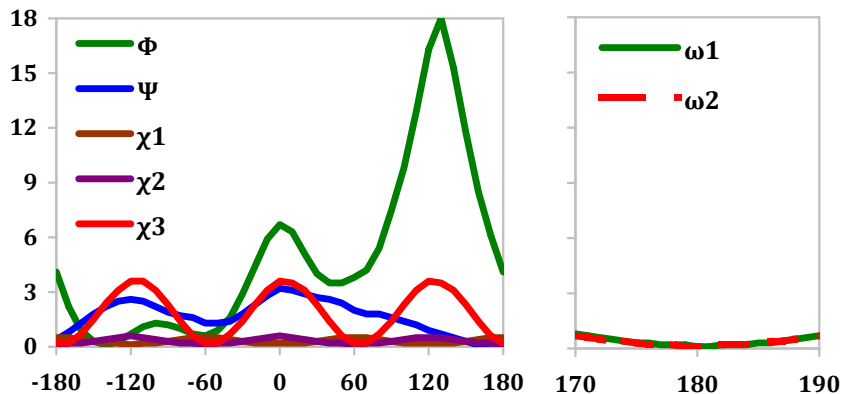


Figure 7.2: One-dimensional projections of the energy landscape of the terminally-blocked alanine for all internal coordinates. For reference, the lowest energy value is set to 0 kcal/mol.

Ala-Nme peptide bond (cf. Fig. 7.1). As the peptide bond torsions are known to undergo only small variations, the ω dihedral angles are allowed to vary only up to 10° from the planar trans conformation.

The ϕ and ψ dihedral angles of the terminally-blocked alanine are very important because their flexibility allows internal hydrogen bonds to form. Consequently, the energy landscape of the terminally-blocked alanine is often projected on these two coordinates to facilitate analysis [32,125,154]. To verify that no important information is discarded by this simplification, we examine other projections of the energy landscape. First, we compute the 1-dimensional projections of the landscape for all coordinates $\{\phi, \psi, \chi_1, \chi_2, \chi_3, \omega_1, \omega_2\}$ (cf. Fig. 7.2). For each coordinate C , its energy profile is obtained by varying the value of the dihedral angle with a 1° step and finding the lowest-energy conformation corresponding to each value. This local minimum is determined using a random restart procedure performing multiple, independent minimizations. Each minimization starts from a randomly sampled conformation, and generates new conformations by perturbing any dihedral angle (except C) of the current minimum. A new conformation is accepted only if its energy is lower than the current minimum. The minimization stops when a maximum number of consecutive rejections is reached.

Fig. 7.2 shows that the dihedral angles having the greatest influence on the shape of the energy landscape of the terminally-blocked alanine are ϕ , ψ and χ_3 . Local minima are located, on the energy profile of ϕ , at -141° , -62° and 44° , on the profile of ψ , at -53° and 163° , and on the profile of χ_3 , at -175° , -55° and 65° . Regarding the other dihedral angles, local minima are located, on the profile of χ_1 , at -120° , 0° and 120° , on the profile of χ_2 , at -60° , 60° and 180° , and on the profiles of ω_1 and ω_2 , at 180° . It also appears from Fig. 7.2 that each coordinate in $\{\chi_1, \chi_2, \chi_3, \omega_1, \omega_2\}$ is likely to be independent from the other six coordinates: the corresponding energy profiles are very smooth, and cyclic in the case of $\{\chi_1, \chi_2, \chi_3\}$. This is not the case for the ϕ and ψ coordinates.

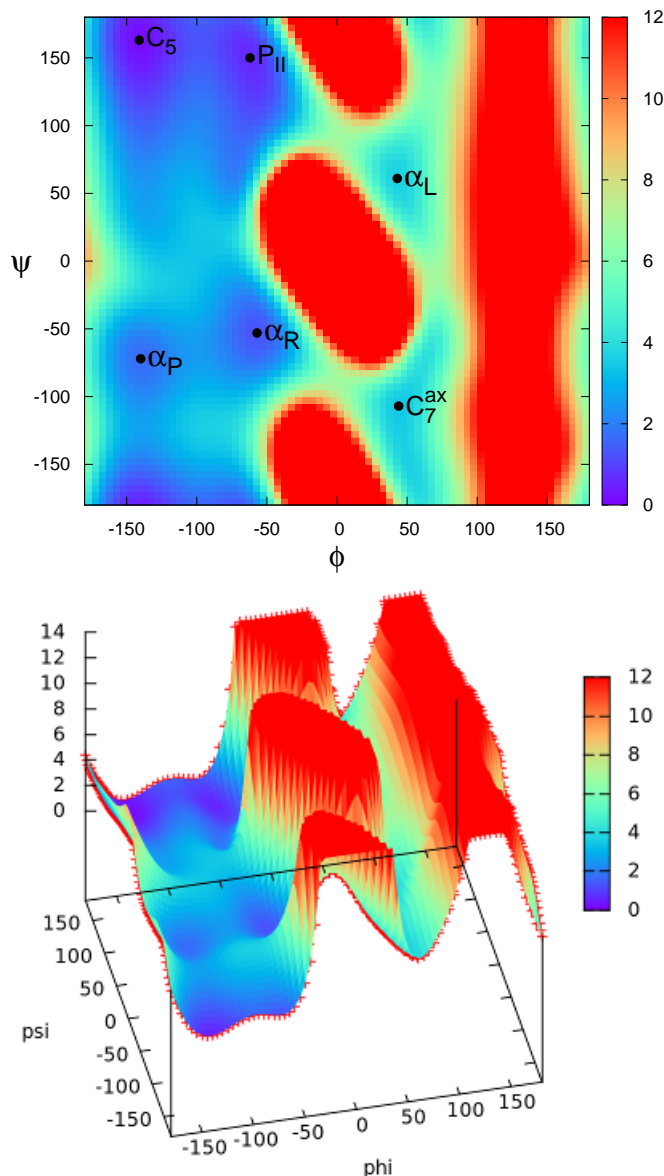


Figure 7.3: Projection of the energy landscape of the terminally-blocked alanine on the (ϕ, ψ) internal coordinates, as seen in 2-D (top) and in 3-D (bottom). For reference, the lowest energy value is set to 0 kcal/mol. Names reported for the local minima are taken from [32, 125].

As ϕ and ψ are not independent, to get a better picture of the energy landscape of the terminally-blocked alanine, we compute a Ramachandran map by projecting this landscape on the 2-dimensional space defined by the (ϕ, ψ) coordinates (cf. Fig. 7.3). This 2-D map is generated by varying both dihedral angles with a 5° step and finding the lowest-energy conformation corresponding to each (ϕ, ψ) pair using a random restart procedure performing multiple, independent minimizations while blocking the (ϕ, ψ) angles. Furthermore, an analysis of the 2-dimensional projections of the landscape associated with the (ϕ, χ_3) and (ψ, χ_3) pairs of coordinates (not shown here) confirm that χ_3 is independent from ϕ and ψ . Therefore, despite its high influence on energy, χ_3 is disregarded in the sequel. The six local minima visible on the (ϕ, ψ) Ramachandran map (cf. Fig. 7.3) correspond to six known meta-stable states of the terminally-blocked alanine, namely the C_5 , P_{II} , α_R , α_P , C_7^{ax} , α_L states [32, 125]. To determine the energy values and the (ϕ, ψ) coordinates of these minima, the exhaustive exploration of the (ϕ, ψ) projection of the energy landscape is refined with a 1° step in the relevant areas. Results are reported in Table 7.1.

Table 7.1: Energy and (ϕ, ψ) coordinates of the local minima of the terminally-blocked alanine. Names are taken from [32, 125]. For reference, the lowest energy value is set to 0 kcal/mol.

	C_5	P_{II}	α_R	α_P	C_7^{ax}	α_L
ϕ ($^\circ$)	-141	-62	-57	-140	44	43
ψ ($^\circ$)	163	150	-53	-72	-107	61
E (kcal/mol)	0	0.44	1.11	1.52	3.27	3.48

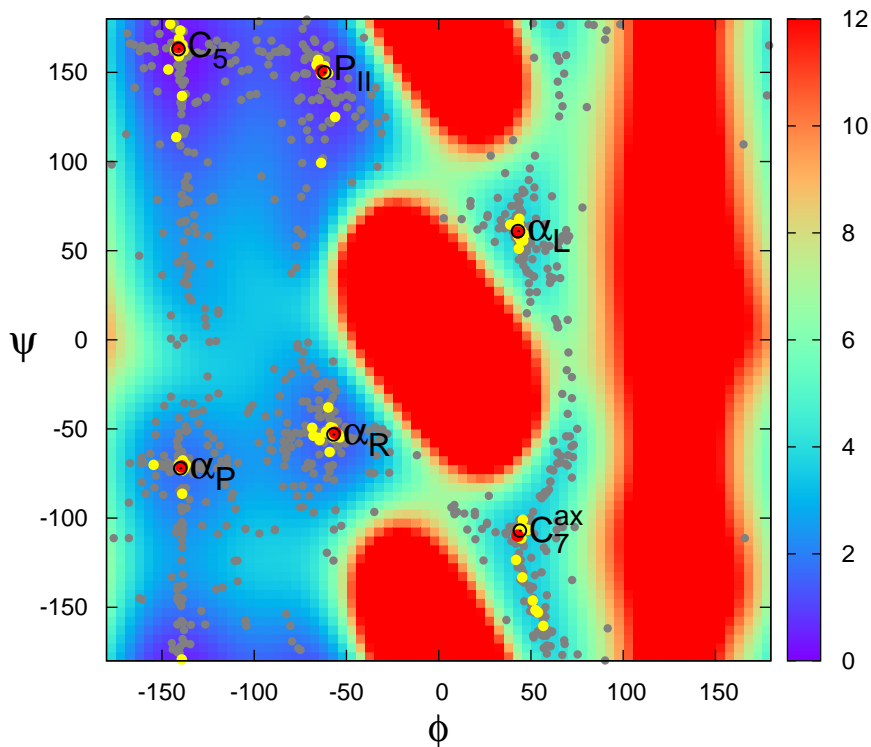


Figure 7.4: Results of the exploration of the energy landscape of the terminally-blocked alanine produced by Basin Hopping. The minima generated by the Monte-Carlo-based minimizations are represented by grey discs, and the selected milestones by yellow discs. Among them, the red discs are the cluster representatives. For reference, the labeled black circles show the six (target) minima identified by the exhaustive search.

Meta-stable States of the Terminally-blocked Alanine

We now show that, using Basin Hopping, we can quickly find the energy minima of the terminally-blocked alanine, as identified by the (costly) exhaustive exploration of the (ϕ, ψ) space presented previously. The parameters of BH are set as follows: we perform 100 rounds, each round executing 10 MC-based minimizations whose maximal number of consecutive rejections is set to 10. Therefore, every run of BH produces 1000 minima and 100 milestones, in about 2 s (on average over 100 runs). The large-amplitude perturbations initiating MC-based minimizations affect only the ϕ or ψ dihedral angle. The small-amplitude perturbations performed during an MC-based minimization affect only one, randomly chosen angle among all dihedral angles. The clustering procedure used to group the milestones is based on an RMSD similarity measure involving the coordinates of the carbon atoms of the backbone of the terminally-blocked alanine, and on a similarity threshold of 0.25 Å. This method consistently produced six clusters, after each one of the 100 runs, and the cluster representatives fitted well the six target minima (see Fig. 7.4). To evaluate the precision of the method, we compute

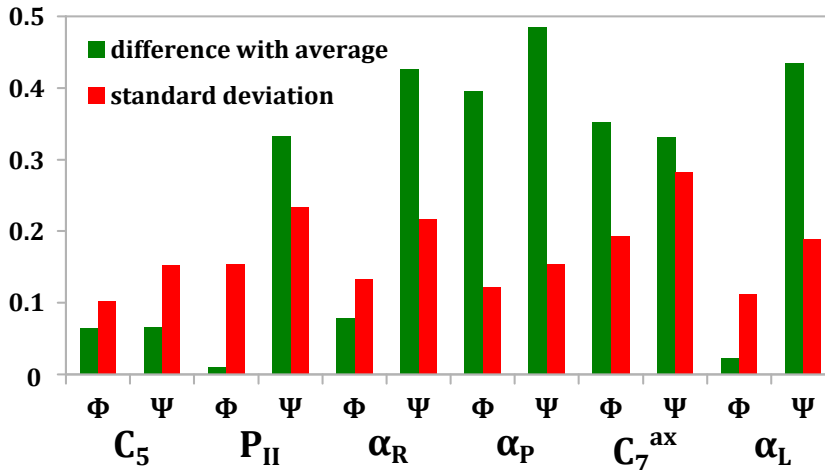


Figure 7.5: Differences between the target minima of the terminally-blocked alanine and the minima returned by BH over 100 runs. For each minimum, and for both ϕ and ψ coordinates, the difference between the target value and the average of the 100 values returned by the BH runs is evaluated, as well as their standard deviation.

the average (and standard deviation) of the coordinates of these representatives, and compare them to the (ϕ, ψ) coordinates of the target minima. Results are reported in Fig. 7.5, and show that precision is very high. For each coordinate of any minimum, the difference between the average of the values returned by BH and the target value is never greater than 0.5° (which is the error inherent to the resolution of the exhaustive search), and the standard deviation of the values returned by BH is never greater than 0.3° .

We have also used BH to obtain a description of the catchment basins on the energy landscape of the terminally-blocked alanine. This enables us to characterize the low-energy basins to which the meta-stable states belong, as well as the transition state ensembles between them. To do that, the parameters of BH are set as follows: we perform 100 rounds, each round executing 10 MC-based minimizations whose maximal number of consecutive rejections is set to 100. Therefore, a run of BH produces 1000 minima, which takes about 25 s. Clustering these minima (and not the milestones, as previously done) produces again the same six clusters. This allows grouping the associated starting points of the MC-based minimizations into six clusters, representing six catchment basins (see Fig. 7.6). A visual analysis of the result shows that these clusters provide a good characterization of the low-energy basins (where the clusters do not overlap) and of the transition state ensembles between them (where the clusters overlap). These regions could be automatically characterized using segmentation techniques, but this is out of the scope of this thesis.

Transition Path Ensembles of the Terminally-blocked Alanine

In this section, we show that the variants of T-RRT can quickly produce many transition paths between the meta-stable structural states of the terminally-blocked alanine. The conformations used as input for the (Anytime) Multi-T-RRT are the energy minima produced by BH. The conformational distance is based on the L^2 norm in the (ϕ, ψ) space. The extension step-size δ is set to 0.1 rad, meaning that the maximal angular variation between two conformations is about 5.7° . The Boltzmann constant being $1.987 \cdot 10^{-3}$ kcal/mol/K, by setting the initial temperature to 70 K, we impose the probability of accepting an energy increment of 0.1 kcal/mol to be around 50% at the beginning of the exploration. The (relative) energy threshold E_{\max} in the transition test is set to 13 kcal/mol. Finally, the temperature increase rate T_{rate} is set to 0.1.

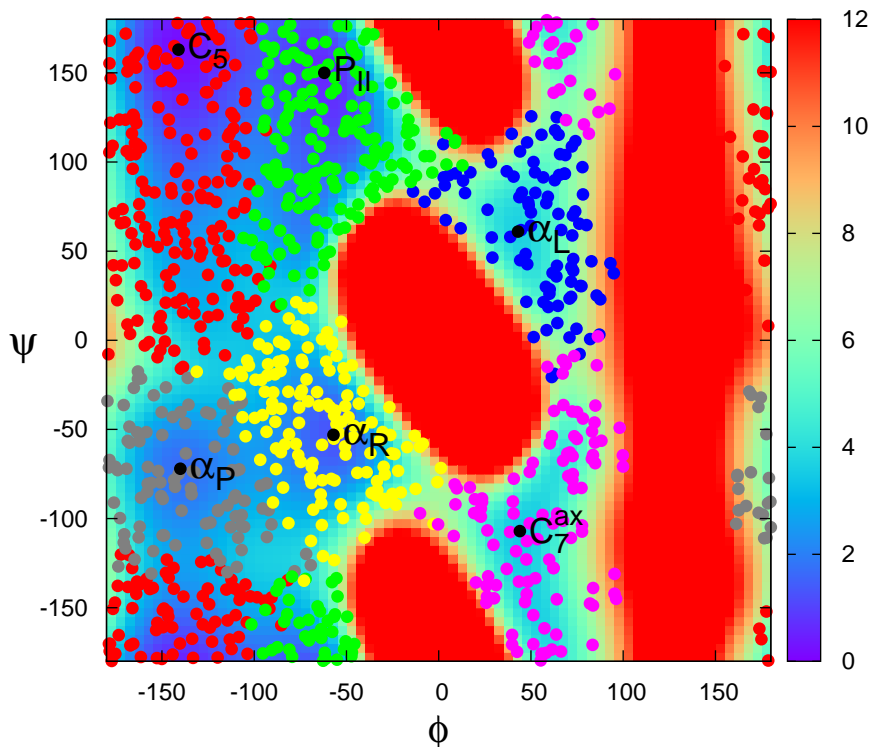


Figure 7.6: Characterization of the catchment basins on the energy landscape of the terminally-blocked alanine, obtained with Basin Hopping. The colored disks represent the starting points of the MC-based minimizations. The color of a disk identifies the cluster to which the minimum produced by the corresponding MC-based minimization belongs.

Starting with six trees, the Multi-T-RRT quickly returns a single tree (containing about 300 nodes, obtained after about 1,500 expansions, in about 0.1 s on average over 100 runs) from which paths connecting the minima are extracted. These paths are then projected on the (ϕ, ψ) space and plotted on the Ramachandran map of the terminally-blocked alanine. We observe that some of these paths may sometimes look “unnatural”: in the instance illustrated by Fig. 7.7 (top), the transition path from P_{II} to α_L goes through almost all the other minima. Using the Anytime Multi-T-RRT allows solving this issue, thanks to the alternative paths in the graph. For example, Fig. 7.7 (bottom) shows a graph obtained after letting the Anytime Multi-T-RRT run for 100 s. This method produces more direct transition paths between all pairs of minima. In fact, the longer the Anytime Multi-T-RRT runs, the more direct paths are found thanks to the increasing coverage of the space by the graph and to the increasing number of cycles in the graph. While remaining acceptable for a user, a running time of 100 s allows obtaining a reasonably dense coverage of the space (i.e. about 6,000 nodes).

To get an idea of the diversity of the transition paths of the terminally-blocked alanine, we run both variants of the Multi-T-RRT 100 times and aggregate all the extracted paths on the same Ramachandran map, as illustrated by Fig. 7.8. We can see that most low-energy and medium-energy regions of the landscape are covered by the transition paths. However, in the case of the Multi-T-RRT, the paths going through the medium-energy area corresponding to $\phi \in [-100^\circ, -50^\circ]$ and $\psi \in [0^\circ, 50^\circ]$ are under-represented in comparison to the case of the Anytime Multi-T-RRT. Finally, we observe that no transition path goes through the energetic barrier corresponding to $\phi \in [100^\circ, 150^\circ]$.

The Anytime Multi-T-RRT can be used to compute transition probabilities between all pairs of energy minima of the terminally-blocked alanine. This is done by running it 1000 times and counting how many runs yield a graph from which a direct transition path can be extracted between a given pair of minima. More precisely, we consider that a run produces a

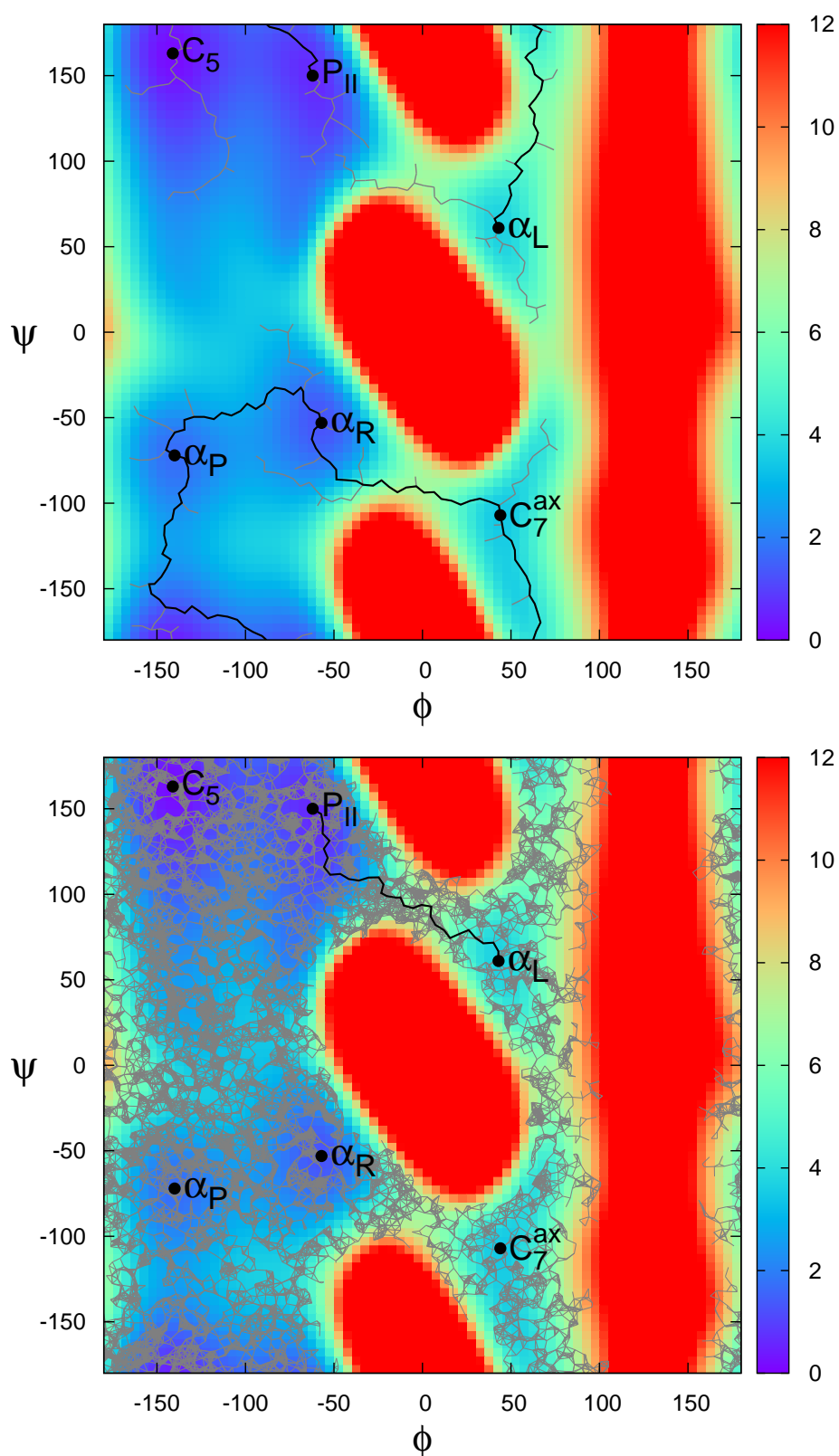


Figure 7.7: Graphs generated by two variants of T-RRT exploring the conformational space of the terminally-blocked alanine. Top: graph produced by the Multi-T-RRT in about 0.1 s. Bottom: graph produced by the Anytime Multi-T-RRT after running for 100 s. The edges of the graph are represented by thin grey lines. The transition path from P_{II} to α_L (extracted from the graph) is highlighted in black.

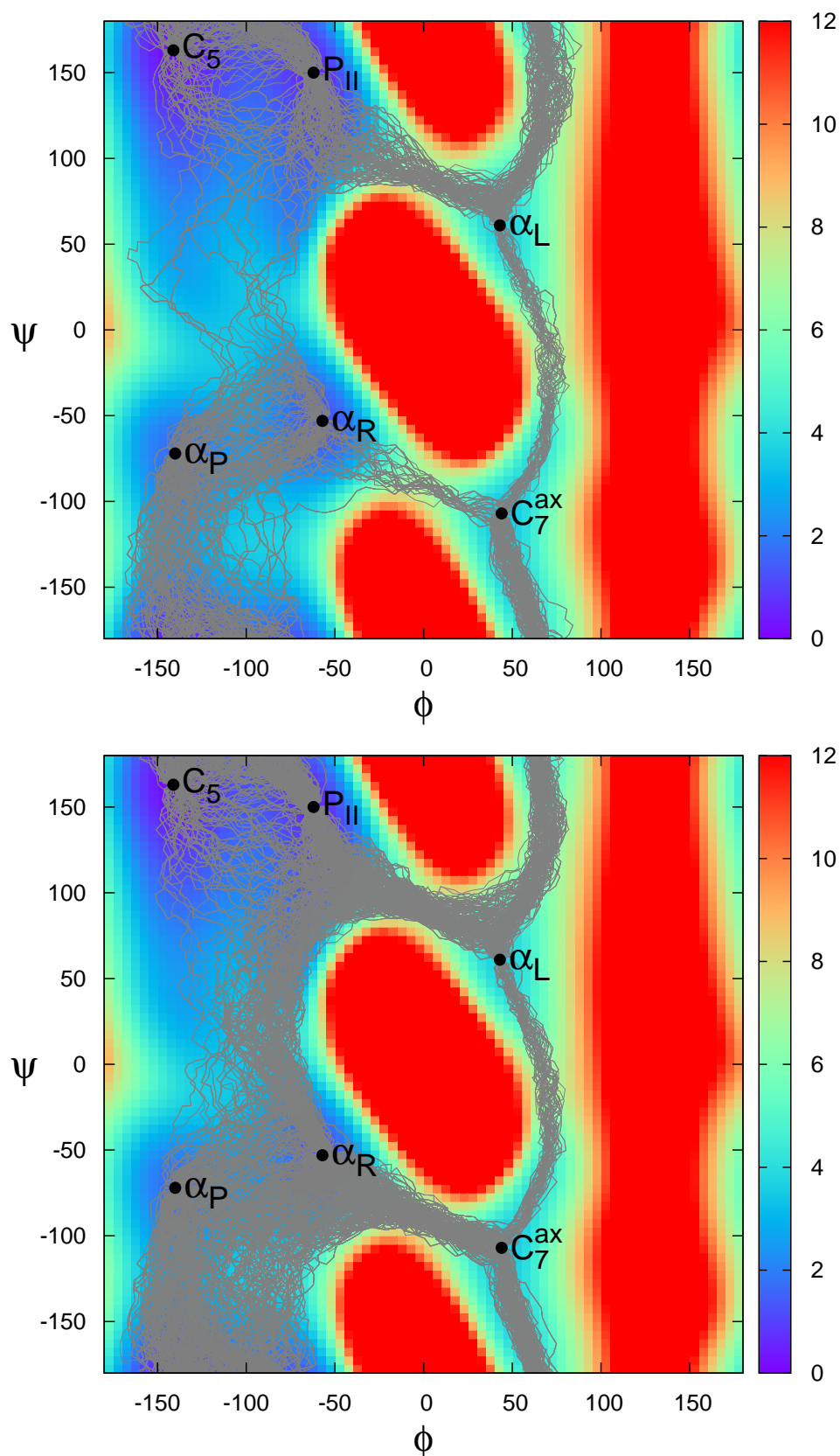


Figure 7.8: Transition paths generated by 100 runs of the Multi-T-RRT (top) and 100 runs of the Anytime Multi-T-RRT (bottom) exploring the conformational space of the terminally-blocked alanine. The maximal running time was set to 100 s for the latter.

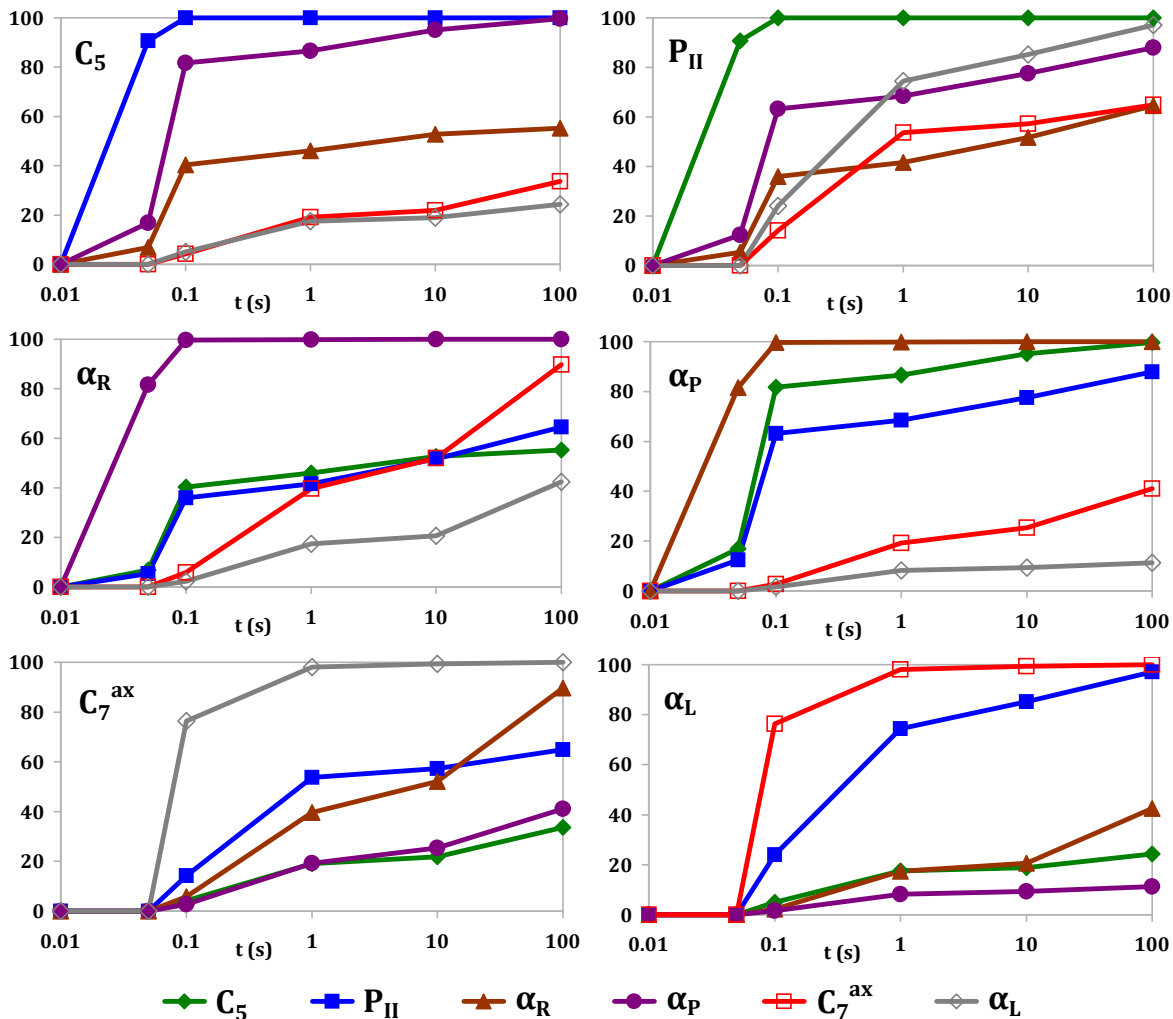


Figure 7.9: Transition probabilities (expressed as percentages) between all pairs of minima of the terminally-blocked alanine, in relation to the running time of the Anytime Multi-T-RRT (in seconds, on a logarithmic scale). Probabilities are computed on the basis of 1000 runs. In each plot, the title indicates the initial conformation of the transition paths extracted from the graph, and the five curves correspond to the possible goal conformations (see color legend).

direct transition path between two minima if the best path in the graph between them does not go through another minimum. By varying the maximal running time of the algorithm, it is possible to determine which transitions are most or least likely. The results of this experiment are reported in Fig. 7.9. It appears that the most likely transitions involve the pairs (C_5, P_{II}) and (α_R, α_P) , as already shown in [32]. Direct transition paths are easily found between the minima in these pairs, even when runtime is low. If we had no other information about the energy landscape, we could infer that these minima are not separated by high-energy barriers or other low-energy basins. On the other hand, it requires a longer time for the transition probabilities associated with (C_7^{ax}, α_L) to reach 1, meaning that high-energy barriers probably separate them. Furthermore, since transitions from α_P to α_R are very likely, it takes some time before the algorithm is able to produce transition paths from α_P to C_5 or P_{II} that do not go through α_R . That is why the corresponding transition probabilities increase slowly with runtime. This is similar for the transitions going from C_5 to α_P . Interpreting the other curves is difficult: it is impossible to know how many high-energy barriers or low-energy basins separate the corresponding minima without additional information.

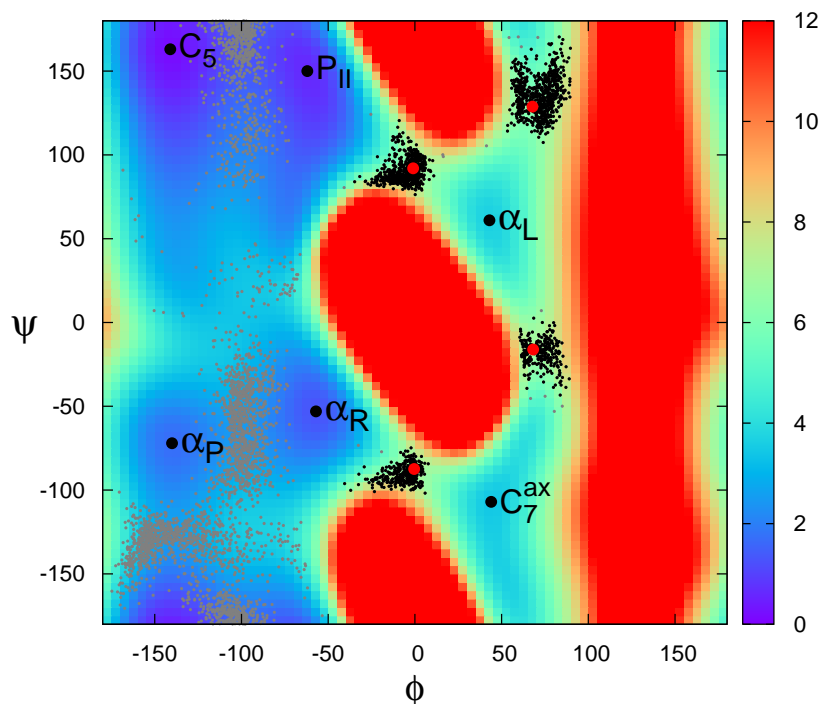


Figure 7.10: Transition state ensembles (represented by black dots) of the terminally-blocked alanine, obtained by aggregating the results of 1000 runs of the Multi-T-RRT, and clustering the extracted transition states (i.e. the highest-energy conformations along all direct transition paths). Transition states rejected by the clustering procedure appear as grey dots. The red disks are the cluster representatives, i.e. the lowest-energy conformations in all clusters.

Transition State Ensembles of the Terminally-blocked Alanine

From the transition paths computed by the Multi-T-RRT, we can extract the transition state ensembles of the terminally-blocked alanine. First, we define a transition state as the conformation having the highest energy along a given path. Our experiment consists of running the Multi-T-RRT 1000 times (which takes about 120 s on average) and aggregating on the Ramachandran map the transition states found along all direct transition paths between any pair of minima. To obtain ensembles, we group the transition states using the same clustering procedure as when grouping the minima produced by BH. Among all the clusters, we reject those containing less than 100 states (i.e. 10% of the number of runs) as being not significant. We also reject clusters containing more than 1000 states because they cannot correspond to a single transition state ensemble. Indeed, as the Multi-T-RRT is run 1000 times, it can produce at most 1000 direct paths between a given pair of minima; therefore, clusters with more than 1000 states encompass several classes of paths (between several pairs of minima). Finally, we choose as cluster representatives the conformations having the lowest energy within each cluster. After repeating the whole procedure 100 times, we observed that it consistently produced four clusters (as illustrated by Fig. 7.10), i.e. four transition state ensembles.

Within each transition state ensemble, the conformation with the lowest energy is chosen as the ensemble representative. To assess the precision of our procedure at determining the (ϕ, ψ) coordinates and the energy values of these representatives, we compute averages and standard deviations across the 100 trials we performed. Results are reported in Table 7.2. From their standard deviations, we can infer that it is difficult to accurately estimate the coordinates of these ensemble representatives. However, since the standard deviation of the energy values is very low (less than 0.05 kcal/mol), we can speculate that this is mainly due to the flatness of the saddles.

Table 7.2: Energy and (ϕ, ψ) coordinates of the transition state ensembles representatives of the terminally-blocked alanine, averaged over 100 trials. Standard deviation is also given for the coordinates. Standard deviation of the energy was less than 0.05 kcal/mol.

ϕ ($^\circ$)	67 ± 3	1 ± 2	0 ± 3	69 ± 3
ψ ($^\circ$)	128 ± 5	92 ± 2	-88 ± 3	-13 ± 4
E (kcal/mol)	6.2	6.8	6.9	7

7.1.3 Conclusion

In this section, we have presented a methodology to explore and obtain a complete representation of the energy landscape of small flexible peptides. The methodology combines a variant of Basin Hopping (BH) with several extensions of T-RRT. By implementing a simplified version of the BH algorithm, where local, gradient-based energy minimization steps are replaced by simple Monte-Carlo-based minimization steps, we achieve an efficient exploration of the energy landscape yielding numerous samples around energy minima. This leads to a quick determination of meta-stable structural states and low-energy basins containing them. In addition, the Multi-T-RRT is very fast at generating transition paths between meta-stable states. This provides us with a good description of the transition path and transition state ensembles. Finally, using the Anytime Multi-T-RRT, we are able to estimate transition probabilities between meta-stable states.

Our objective was to demonstrate that combining sampling-based algorithms such as BH and T-RRT allows quickly obtaining an accurate representation of the energy landscape of a small yet highly-flexible peptide. Studying the terminally-blocked alanine served as a proof-of-concept here. Our next targets are bigger peptides, starting with the met-Enkephalin (Tyr-Gly-Gly-Phe-Met). Despite the great interest it has sparked due to its dynamical structure and its potential role in pain inhibition, alcoholism, and cancer treatment, this peptide is currently poorly understood [140]. The preliminary tests we have performed on the met-Enkephalin show that the methods we propose can remain efficient even with bigger peptides. Indeed, when running the Multi-T-RRT using as input three conformations corresponding to local energy minima of the met-Enkephalin (computed in another context by a simulated annealing procedure), we obtain a graph connecting them in about 0.08 s (on average over 100 runs). As part of our ongoing work, we also have to ensure that the clustering procedures involved in our methods remain effective when dealing with bigger peptides.

Directions of future work include exploiting the graph produced by a single run of the Anytime Multi-T-RRT to describe transition path and transition state ensembles. This will allow us to make better use of computational resources, as opposed to aggregating paths extracted from several runs of the Multi-T-RRT. In addition, Markov-based transition-step analysis can be conducted on the graph produced by one or more runs of the (Anytime) Multi-T-RRT. This analysis can allow estimating stabilities of each computed state, and provide a rigorous basis for the designation of a state as stable or semi-stable.

In contrast to robotics, computational structural biology offers great opportunities to tackle very complex and high-dimensional problems that appear when studying large molecules. In the context of the exploration of energy landscapes, analyzing large molecules means having to deal with numerous interesting structural states, whether local energy minima or local energy saddle-points. This results in extremely challenging settings for the multiple-tree (anytime) variants of T-RRT that we have presented in this thesis. Indeed, we plan to analyze how well these algorithms can scale by growing hundreds or even thousands of trees simultaneously in a molecule's conformational space.

7.2 Simulation of Protein-Ligand Unbinding

In the past, experimental and computational approaches aimed at investigating protein-ligand interactions have mostly focused on the molecular complex, when the ligand is docked in the active site of the protein. However, an increasing amount of research shows that important interactions that may determine protein-ligand specificity and activity occur far away from the active site, during ligand binding or release (see for example [16, 28, 101, 132]).

Despite impressive advances in structural biology techniques, obtaining accurate experimental data about protein-ligand interactions taking place far from the active site remains challenging. Computational methods may help in better understanding such interactions. However, simulating ligand (un)binding, particularly when the active site is deeply buried in the protein, is also a difficult problem for current computational approaches. Some variants of molecular simulation methods have been devised specifically for that. In particular, Steered Molecular Dynamics (SMD) [80] and Random Acceleration Molecular Dynamics (RAMD) [111] have become popular techniques for the simulation of ligand (un)binding. Both methods are based on the same principle: the application of an artificial force to accelerate the ligand diffusion inside the protein. The direction of this force is defined by the user, in the case of SMD, or chosen randomly, in the case of RAMD. Although the interest and the relevance of these methods are not questioned here, it should be noted that this artificial force must be carefully parameterized to avoid strongly biased (or inaccurate) results. Monte-Carlo-based techniques have also been proposed to study ligand (un)binding and diffusion [20]. They perform a more computationally-efficient exploration of the conformational space compared with techniques based on molecular dynamics simulations, and do not require additional artificial forces in the molecular force field to accelerate simulations. Nevertheless, all the aforementioned methods remain computationally expensive.

Our work is based on an original approach to simulate protein-ligand (un)binding, and other types of large-amplitude (long time-scale) molecular motions, at a very low computational cost [37, 39, 99]. This approach involves a mechanistic representation of molecules, and RRT-like algorithms to explore their conformational space. It has been validated on experimental data, and compared to other computational methods. It has also been successfully applied to rational enzyme engineering [73, 101].

This section presents a technique to simulate protein-ligand unbinding in a computationally efficient way. Starting from the model of a protein-ligand complex, this technique computes the ligand unbinding path from the active site to the surface of the protein. In the current version, flexibility is considered only for the ligand and the protein side-chains. Furthermore, only geometric constraints are involved, which means that no molecular energy is computed, and that motions are validated only on the basis of collision avoidance. Our approach involves a variant of RRT, called Manhattan-like RRT (ML-RRT) [36, 37], to explore the conformational space of the protein-ligand complex. Such a purely geometric approach, together with the efficiency of the exploration algorithm, enables the simulation of ligand unbinding within very short computing time, ranging from a few seconds to a few minutes. Achieving such low runtime is a constraint we had imposed on ourselves, with the objective of developing an efficient web server, that we have named MoMA-LigPath¹, as it builds on the MoMA software. Even though they satisfy only geometric feasibility, the paths generated by this approach can provide interesting information to biologists and chemists. They can also serve as a first approximation that can be further refined using standard molecular modeling techniques. In this section, we show the interest of this approach on the hexameric insulin-phenol complex.

Finally, note that our goal in the long run is to take molecular energies into account and to integrate variants of T-RRT into this approach (as discussed in Section 7.2.3).

¹<http://moma.laas.fr>

7.2.1 Methods

Manhattan-like RRT (ML-RRT)

The method we have developed to simulate protein-ligand unbinding features a robotics-inspired algorithm based on the Manhattan-like RRT (ML-RRT) [36, 37]. The ML-RRT algorithm was originally proposed as an extension of RRT for disassembly path planning of complex objects with articulated parts [35]. Its key concept is to divide variables (i.e. conformational parameters, in the molecular context) into two groups, called active and passive variables, and to generate their motions in a decoupled manner. Active variables correspond to parts whose motions are essential for the disassembly task, whereas passive variables correspond to parts that should move only if they hinder the motions of other mobile parts (active or passive). Compared with the basic RRT, the ML-RRT algorithm offers two important advantages. First, it can solve problems that are practically intractable with RRT, by implicitly reducing the dimensionality of the search space. Second, it allows automatically identifying which parts should move to find a solution to the disassembly problem.

In the structural biology context, ML-RRT considers the ligand as an articulated mechanism to be disassembled from the protein. All the protein side-chains are also articulated with freely rotatable bonds. For the conformational exploration, the active variables are the parameters defining the pose (position and orientation) of a reference frame associated with the geometric center of the ligand, as well as the ligand bond torsions; the passive variables are the bond torsions of the protein side-chains. Collision avoidance between non-bonded atoms is the only feasibility condition evaluated when validating molecular motions.

Parameters of the Approach

The main input of MoMA-LigPath is a *.pdb* file containing the atomic coordinates of the protein-ligand complex. This file can contain several proteins and ligands, as well as other molecules, such as structural waters or ions, but only one ligand is considered to be mobile. Moreover, only the side-chains (and not the backbone) of a molecule can be defined as flexible. By default, all side-chains are considered flexible, but one can locally modify the flexibility of a molecule by blocking some side-chains. The main output of MoMA-LigPath is a set of solution paths. Each solution is a sequence of intermediate conformations of the molecules along the ligand exit-path. The path is discretized in such a way that the maximum displacement of an atom of the ligand between two consecutive frames is approximately 0.5 Å.

The method allows tuning the following parameters:

- % of van der Waals radii: This parameter is used for collision detection between non-bonded atoms. 75% is a reasonable default value, often used to check atom overlaps in other computational methods. A lower percentage may be necessary to find solutions to very constrained problems, which would require some flexibility of the protein backbone. In easy cases, this value can be increased to force the ligand to move along the medial axis of the exit channel.
- RRT expansion strategy: By default, the method applies a Connect strategy to expand nodes during the construction of the search tree. This can be changed to the more basic Extend strategy, which generally produces shorter local moves, and is thus more computationally expensive. Nevertheless, the Extend strategy can be more efficient when solving very constrained problems.
- Exit distance: The length of the unbinding paths to be produced can be specified by defining the distance (in Å) that has to be reached between the geometric centers of the ligand and the protein. When left unspecified, this distance is computed automatically, based on the molecule sizes.

- N fail max: This parameter determines the number of consecutive expansion failures after which a node in the search tree is considered “exhausted”, and is no more selected for expansion during the ML-RRT construction. This heuristic is further explained in basic papers on ML-RRT [36]. The default value, 50, provides good results in general. This value can be increased for very constrained problems.

7.2.2 Results and Discussion

We now briefly present some results we have obtained with our unbinding simulation method for the hexameric insulin-phenol complex. This molecular complex is an interesting test system because of the likely existence of multiple pathways for phenol unbinding. The reported results only aim to illustrate the capabilities of the method, further biological interpretations being out of the scope of this thesis. More detailed explanations on the application of robotics-inspired algorithms to simulate protein-ligand unbinding in the context of enzyme engineering can be found in [73, 101].

Insulin, in its monomeric, active form, is composed of two short peptide chains. In the presence of zinc ions, insulin monomers tend to associate, forming more stable hexameric structures [55]. Different conformational states of the insulin hexamer have been observed experimentally. Here, we consider the so-called R_6 state, which has a threefold symmetric, toroidal shape (see Fig. 7.11), and is stabilized by bound phenolic molecules [11]. Understanding the mechanism of phenol unbinding is important because it is possibly involved in the conversion of the hexamer into the monomeric, active form of insulin [153]. Note that the study of the hexameric insulin-phenol complex and of the hexamer-monomer conversion are of interest in pharmacology for the treatment of type 1 diabetes.

The structure of the R_6 insulin hexamer, determined by X-ray crystallography, is available in the Protein Data Bank², with PDB ID: 1ZMJ. The R_6 insulin hexamer presents six hydrophobic pockets containing bound phenol molecules. We have applied our method to simulate the unbinding of one of them: the one located in the binding pocket between chains A, B, F and H. The other phenol molecules were kept in the input file, and considered to be static molecules. The side-chains of the six histidines interacting with the zinc ions were blocked. Hydrogen atoms were not added. This is acceptable because the solutions provided by our method are not expected to be an accurate representation of unbinding paths, but simply a first approximation.

As a first experiment, we simulated 20 phenol unbinding paths. To emphasize the computational efficiency of the method, let us mention that the average computing time for one solution was less than 10 s on a single processor. A significant variability in the solutions can be observed. In most simulations, the phenol molecule exits following pathways at the interface between chains A, F and H (see Figure 7.11). Such paths can be clustered into two groups, which we refer to as pathway 1 (PW1) and pathway 2 (PW2), following the notations used in related work [153]. 25% of the paths (5 over 20) follow PW1. The ligand finds a passage between residues Ile10-A and His5-F, by inducing a significant conformational change of His5-F. Note that ring-flipping of His5 has been suggested by other computational and experimental studies on this system [153]. The ligand also induces the motion of the side-chain of Tyr16-H, which itself induces the motion of Tyr26-F. The conformations of other residue side-chains are also slightly perturbed by phenol unbinding following PW1.

PW2 is the most frequent pathway observed among the 20 solutions: 60% of the paths follow PW2. After a quick analysis of the solutions using a molecular viewer, one can observe that PW2 is the shortest and geometrically easiest pathway between the binding pocket and the surface of the protein, which explains the highest probability to obtain this type of solution. The ligand follows a narrow, partially open channel, and induces slight conformational changes

²<http://www.rcsb.org/pdb>

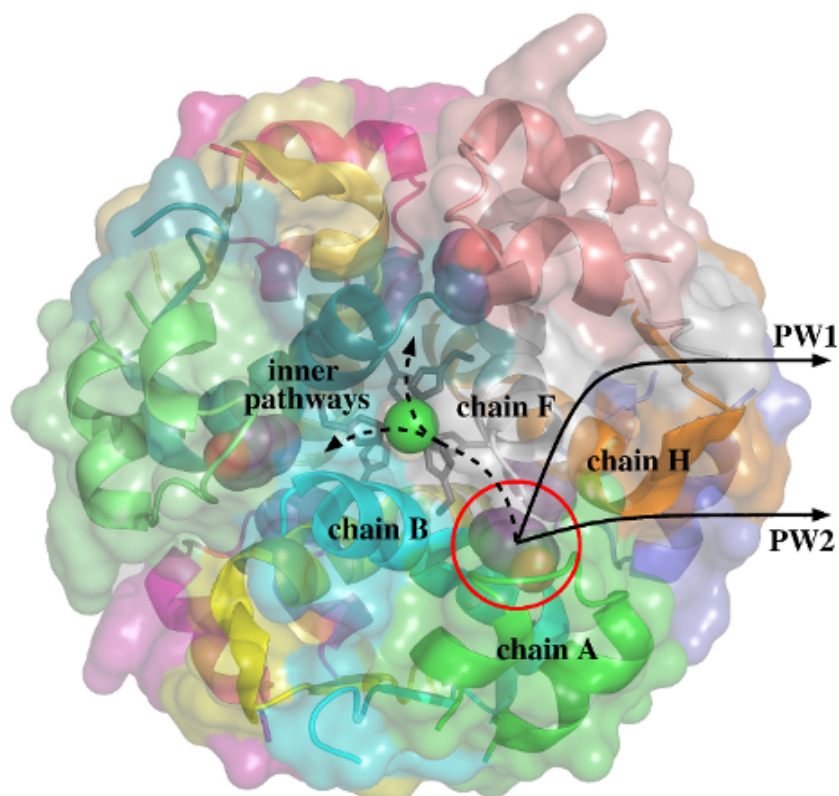


Figure 7.11: Structure of the R_6 hexameric insulin-phenol complex. The phenol molecule in the pocket between chains A, B, F and H can follow different unbinding pathways. The two most likely pathways are located at the interface of chains A, F and H. However, diffusion through the inner part of the hexamer is also geometrically feasible. Images of molecular models in this section have been generated using PyMOL (<http://www.pymol.org>).

of only a few residues, mainly of Leu17-H. Remarkably, a combination of RAMD and SMD methods also pointed out PW2 as the most likely pathway for phenol unbinding [153].

Fig. 7.12 shows some intermediate frames of two solutions: one following PW1 and the other following PW2. The ligand and moving side-chains are represented. The figure also illustrates another type of pathway that was not reported in related work [153]: Surprisingly, in a few simulations (3 over 20), the ligand diffuses inside the insulin hexamer before finding an exit pathway. Indeed, the phenol molecule moves at the interface of insulin monomers toward the center of the hexamer, as indicated by the dashed line in Fig. 7.11, eventually finding exit channels that are the symmetric counterparts of PW1 and PW2. Note that the other phenol molecules, considered to be static in the simulations, do not obstruct these pathways. In two of the simulations, the ligand exits through a pathway similar to PW2 between chains F, H and K. One of such inner pathways is represented in Fig. 7.12. In another simulation, the ligand follows a pathway similar to PW1 between chains C, J and L. Exit through all the other pathways symmetric to PW1 and PW2 seems to be geometrically feasible.

Finally, note that none of the 20 simulation runs reported here, and none of 100 additional runs performed in the same conditions (as a second experiment), was able to find a third class of pathway (PW3) obtained by RAMD simulations, as reported in [153]. PW3 is located at the interface between the two chains of the insulin monomer. It is a narrow corridor, involving steric interactions of phenol with many residues. The fact that PW3 was not found means that it is geometrically unlikely, or even impossible, if the protein backbone does not deform. In a third experiment, we have run 20 additional simulations with a reduced atom size, namely

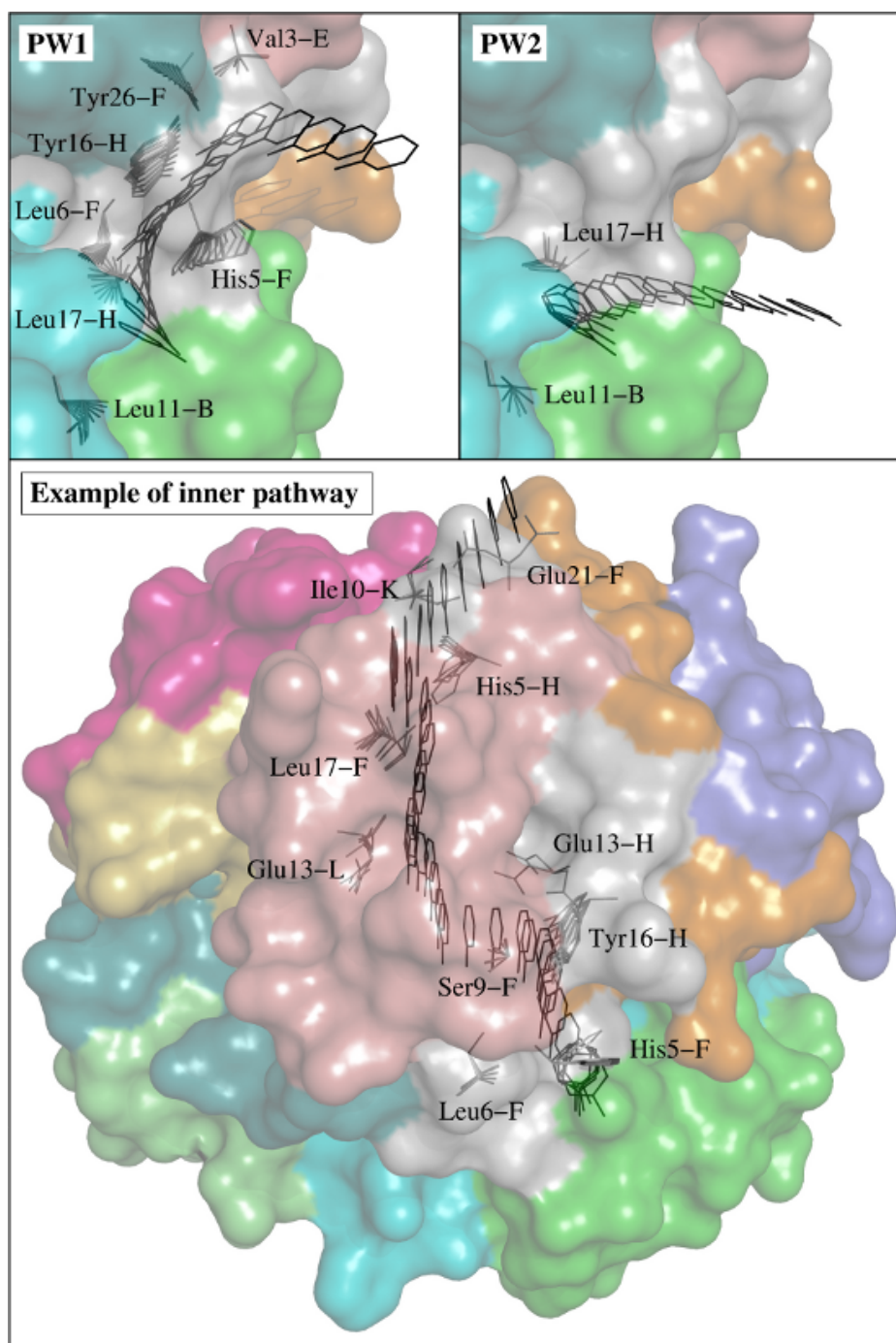


Figure 7.12: Different paths for phenol unbinding from the R_6 insulin hexamer. The location of the phenol molecule and the conformations of moving side-chains are represented for some intermediate frames. The two images at the top correspond to paths following the most likely unbinding pathways: PW1 and PW2. The image at the bottom illustrates one of the pathways going through the inner part of the insulin hexamer.

60% of van der Waals radii, to simplistically emulate slight fluctuations of the backbone. In this case, 10% of the solutions followed PW3. Nevertheless, since some of the results obtained with such a reduced atom size can be unrealistic, we prefer not to argue about the existence of this pathway type. More generally, a further analysis of the results presented here, considering molecular energies, would be necessary to yield a more accurate model of phenol unbinding from the R_6 insulin hexamer.

7.2.3 Conclusion

In this section, we have presented a method to simulate protein-ligand unbinding in a computationally efficient way. This efficiency has enabled us to develop a web server, named MoMA-LigPath, which is, to the best of our knowledge, the first such application. Our approach is based on a mechanistic representation of the molecular system, considering partial flexibility, and on the application of the ML-RRT algorithm to explore the conformational space. The simplicity of the molecular model, together with the efficiency of the exploration algorithm, allow for the simulation of protein-ligand unbinding within very short CPU time, ranging from a few seconds to a few minutes. We have observed that the performance of the method depends on the geometric difficulty, which is mainly related to the narrowness/topology of the access/exit channel, rather than on molecule sizes.

The simulation method we have proposed can be interesting for protein engineering, to help decision making for site-directed mutagenesis experiments. More generally, information on local protein-ligand interactions taking place far away from the active site may help understanding the overall molecular interaction mechanism. Nevertheless, although geometry can play a decisive role, it is well known that electrostatics, pH and other conditions that are not considered in this work can also dramatically affect protein-ligand (un)binding. Therefore, solutions provided by our method have to be considered as a first approximation that may require subsequent refinement and analysis using more accurate models. Note that this approach has already started to bare fruits: since this work was published [45], MoMA-LigPath has been involved in the (un)binding analysis of an anticancer compound [86].

As part of our future work, we plan to extend the capabilities of our simulation method. For instance, the issue of limited flexibility can be addressed by utilizing an improved version of ML-RRT taking backbone flexibility into account [37]. More importantly, our goal on the long run is to take molecular energies into account. For that, we will integrate in our approach the variants of T-RRT introduced in this thesis. However, they cannot be used as such because T-RRT suffers from the same limitations as RRT on such high-dimensional problems. Recall that the interest of using ML-RRT in this work is that it allows implicitly reducing the dimensionality of the conformational space. Therefore, if we want to maintain computational efficiency, we will have to combine ML-RRT with the variants of T-RRT we have proposed in previous chapters, as was done for the basic T-RRT itself in [79].

Chapter 8

Conclusion

In this thesis, we have addressed the issues of improving the efficiency and enhancing the capabilities of sampling-based algorithms in the context of feasible, cost-space, and optimal path planning. Our intention was to deal with difficult, high-dimensional problems involving complex, continuous cost functions. Therefore, our work has more specifically focused on cost-space and optimal path planning. This was justified by the fact that, even though classical sampling-based methods have reached a high level of success, their cost-space counterparts have been developed only recently and still require further research. In Chapters 3 to 5, we have introduced several extensions of RRT-like path planners, based on the T-RRT and RRT* algorithms. In Chapters 6 and 7, we have presented various applications of these novel approaches in the fields of robotics and computational structural biology.

8.1 Summary of the Algorithmic Contribution

First, we have proposed several extensions of the T-RRT algorithm in order to enhance cost-space path planning [49]. More precisely, we have improved its mono-directional variant by optimizing its transition test. We have also proposed a bidirectional variant of T-RRT, taking into account the cost constraints involved. We have shown that, in the context of a simple “init-to-goal” path planning problem, using the Bidirectional T-RRT is a better strategy than using the Connect or Goal-biased mono-directional T-RRT. Later on, as a generalization of this bidirectional approach, we have proposed a multi-tree variant of T-RRT that can compute a path going through a given set of waypoints [52].

Second, we have explained how to solve optimal path planning problems in a more efficient way than what RRT* currently allows on cost spaces. By combining the underlying concepts of T-RRT and RRT*, i.e. a cost-based node creation and a quality-based edge management, we have proposed two new sampling-based algorithms that are asymptotically optimal: the Transition-based RRT* and the Anytime T-RRT. We have also shown that they converge faster than RRT* toward the optimal path, especially when the topology of the configuration space is complex and/or when its dimensionality is high [51].

Third, we have proposed three parallel versions of RRT targeting large-scale distributed-memory architectures: the OR parallel RRT, the Distributed RRT, and the Manager-worker RRT [48]. We have also discussed how these parallelization schemes can be applied to other RRT-like algorithms such as those introduced in this thesis. We have observed that using these parallel algorithms can significantly reduce running times when dealing with complex robotic problems or structural biology problems [50].

Finally, we have suggested that combining some of these novel approaches can allow for new kinds of planning problems to be solved. For instance, by integrating the Anytime T-RRT and the Multi-T-RRT, we have developed an Anytime Multi-T-RRT that we have utilized in several applications.

8.2 Summary of the Applications

In robotics, we have first utilized the Anytime Multi-T-RRT to solve ordering-and-pathfinding problems [52]. With this approach, such problems can be solved from a purely geometrical perspective, without having to use a symbolic task planner. As an example, we have presented a simulated industrial inspection problem involving an aerial robot. Second, we have explained how a cost-space approach can be useful to perform precise 6-dimensional manipulation tasks with the FlyCrane, a towed-cable system involving three aerial robots [115, 116]. For that, we have defined an application-specific configuration-cost function taking the constraints of this robotic platform into account. We have evaluated the approach, in simulation, on 6-D manipulation problems involving this platform.

In computational structural biology, we have used the (Anytime) Multi-T-RRT, in addition to another sampling-based method called Basin Hopping, to produce a full characterization of the energy landscape of a small flexible peptide [47, 53]. This combined approach allowed for the determination of meta-stable structural states of the peptide, as well as transition state ensembles, transition path ensembles, and transition probabilities between these meta-stable states. We have validated this approach on the terminally-blocked alanine. Another application we have presented here is the simulation of the unbinding process of a protein-ligand complex [45, 46]. As a first step, we have proposed a geometric approach involving the Manhattan-like RRT, that we could implement as a web server, thanks to its very short running times. We have demonstrated the interest of this method on the hexameric insulin-phenol complex.

8.3 Directions of Future Research

As an initial direction for future research, we could start by improving the methods presented in this thesis. Indeed, some of our approaches can be investigated further to reach higher levels of efficiency. First, in the cost-space path-planning context, T-RRT-like algorithms can be enhanced by taking the configuration-cost function into account in a more effective way. Indeed, a drawback of the rejection-sampling strategy currently in use is its wastefulness. A possible solution is to partially guide the exploration of the cost landscape by exploiting, for example, gradient-based methods, as is done in [12]. However, this requires maintaining a good balance between the resulting bias and the inherent exploratory strength of RRT-like algorithms. Second, in the optimal path-planning context, we can develop a more efficient path planner based on AT-RRT, T-RRT*, or a combination of both. This requires analyzing AT-RRT and T-RRT* further, and determining which strategy works best in general or on specific classes of problems. Finally, in the parallel path-planning context, a more powerful parallelization scheme can be obtained by combining the three parallel approaches presented in this thesis.

On another front, we should consider taking full advantage of the graph produced by the anytime variants of T-RRT-like methods, as this can have beneficial repercussions. First, on the computational structural biology side, when exploring the energy landscape of a peptide, we can exploit this graph to describe transition path and transition state ensembles, and to estimate transition probabilities between meta-stable states. This would make better use of computational resources than the current method does, and allow us to study bigger peptides. Second, on the robotics side, instead of computing a path going through all the waypoints while considering them as having a similar status, we can envision other useful applications. For example, we can solve problems involving a single initial configuration and several potential goal configurations. The different paths extracted from the graph can represent several ways to solve this hybrid task-and-path planning problem, allowing us to choose the best path or to change the current plan.

In robotics, combining the methods we have proposed in this thesis offers great possibilities. For example, in the context of 6-dimensional manipulation with the FlyCrane, the approach can be enriched by using some of the more sophisticated variants of T-RRT. This is particularly true for the Anytime T-RRT, as it would allow us to produce the optimal path to perform a given manipulation task. Furthermore, the Multi-T-RRT can be used to solve complex task-and-path planning problems encompassing the manipulation of several objects for the (dis)assembly of a large structure.

Some of the algorithms we have introduced can be used in other planning contexts than those studied here. First, anytime variants of T-RRT-like methods can be useful for path replanning. As they build a graph containing cycles, they provide alternative paths over the space, that are readily available if the current solution path is invalidated because of errors in the model or moving obstacles. Second, these anytime algorithms can be used for online planning. While only part of the current solution path is executed by a robot, the rest of the path can be further optimized [94].

In computational structural biology, interesting new problems can be solved by using the algorithms presented in this thesis. This is particularly true for the simulation of protein-ligand interactions because we have not applied the multi-tree and anytime paradigms to this problem yet. First, if various conformations of a protein-ligand complex are available, a multiple-tree approach can allow us to generate several possible unbinding pathways at the same time, and to determine the most likely one. Second, an anytime approach would allow us to find the optimal unbinding pathway with respect to a given path-quality criterion, such as minimum resistance or maximum flux.

The structural biology domain produces examples potentially requiring enormous amounts of computational resources. When exploring the conformational space of a molecule or of a molecular complex, growing several trees simultaneously has already proven useful for the applications presented in this thesis. Going even further, instead of building a few trees on the search space, it can be useful to grow hundreds or even thousands of trees simultaneously. The parallel-planning paradigm can allow the multiple-tree variants of T-RRT-like methods to meet this challenge, by interleaving several levels of parallelization. For instance, we can imagine to combine the three following levels: 1) distributing the construction of the trees over several groups of processes; 2) sharing the construction of each tree between several processes; 3) parallelizing the most computationally-expensive components of the T-RRT expansion. Using such parallel versions of multi-tree algorithms would enable us to exploit numerous computational resources and to solve extremely complex problems.

Appendix A

Extensions des Méthodes de Planification de Chemin par Échantillonnage dans des Espaces de Coût Complexes : Applications en Robotique et en Biologie Structurale

Résumé Abrégé

Planifier le chemin d'un robot dans un environnement complexe est un problème crucial en robotique. Les méthodes de planification probabilistes peuvent résoudre des problèmes complexes aussi bien en robotique, qu'en animation graphique, ou en biologie structurale. En général, ces méthodes produisent un chemin évitant les collisions, sans considérer sa qualité. Récemment, de nouvelles approches ont été créées pour générer des chemins de bonne qualité : en robotique, cela peut être le chemin le plus court ou qui maximise la sécurité ; en biologie, il s'agit du mouvement minimisant la variation énergétique moléculaire. Dans cette thèse, nous proposons plusieurs extensions de ces méthodes, pour améliorer leurs performances et leur permettre de résoudre des problèmes toujours plus difficiles. Les applications que nous présentons viennent de la robotique (inspection industrielle et manipulation aérienne) et de la biologie structurale (mouvement moléculaire et conformations stables).

Mots-clés

planification de chemin · espace de coût · robotique · biologie structurale

Résumé

Planifier un chemin pour un système mobile dans un environnement complexe est un point crucial en robotique. Dans ce contexte, les algorithmes de planification de chemin par échantillonnage ont eu beaucoup de succès malgré leur simplicité conceptuelle. Cela vient de leur capacité à résoudre des problèmes de planification complexes faisant intervenir des systèmes mobiles ayant de nombreux degrés de liberté. Ces algorithmes ont été utilisés dans des domaines aussi divers que la robotique, l'aérospatiale, le prototypage virtuel, l'animation graphique, la biologie structurale, et la médecine. Leur principe de base est d'explorer l'espace des configurations du système mobile en l'échantillonnant, et de construire un graphe représentant la topologie de cet espace.

En général, la planification par échantillonnage produit des chemins faisables (ou sans collision) mais ne considère pas leur qualité. On parle de "planification faisable". Cependant, dans de nombreux domaines, il est important de calculer des chemins de bonne qualité sur la base d'un critère donné. Récemment, des variantes des méthodes de planification classiques ont été créées pour prendre en compte une fonction de coût durant l'exploration. Cela s'appelle la "planification en espace de coût". Certaines méthodes visent même à trouver le chemin optimal dans cet espace de coût. On parle alors de "planification optimale".

Dans cette thèse, nous proposons plusieurs extensions de méthodes de planification par échantillonnage, pour résoudre efficacement des problèmes toujours plus difficiles. En effet, de nombreuses applications produisent des problèmes en haute dimension avec des contraintes géométriques et différentielles complexes. Les méthodes existantes ne peuvent pas résoudre de tels problèmes, ou difficilement. Notre travail est centré sur des applications en robotique et en biologie structurale. En robotique, nous développons des méthodes de planification pour des robots évoluant dans de très larges environnements, tels que des installations industrielles. Au-delà du simple calcul d'un chemin allant d'un point A à un point B, une inspection industrielle nécessite de calculer un chemin visitant plusieurs points de façon efficace. Un autre problème complexe est la manipulation précise en 6 dimensions exécutée par un système à câbles avec trois robots aériens. En biologie structurale tous les problèmes sont difficiles de par leur dimensionnalité, même pour de petites molécules. Notre travail comporte deux aspects : explorer le paysage énergétique d'un petit peptide, et simuler le processus de séparation d'un ligand et d'une protéine.

Les méthodes présentées dans cette thèse se basent sur le Rapidly-exploring Random Tree (RRT), un algorithme populaire en planification faisable, ainsi que certaines variantes: le Transition-based RRT (T-RRT), pour la planification en espace de coût, et RRT*, pour la planification optimale. La contribution algorithmique de cette thèse est triple. (1) Nous présentons des extensions bidirectionnelles et multi-arbres de T-RRT, pour une planification en espace de coût plus efficace. Nous les appliquons à des problèmes de robotique (inspection industrielle et manipulation aérienne) et de biologie structurale (mouvement moléculaire et conformations stables). (2) Nous combinons les concepts de T-RRT et RRT* de deux façons différentes conduisant à de nouvelles approches améliorant la planification optimale. Nous les évaluons sur plusieurs problèmes de robotique, et montrons que l'amélioration est particulièrement significative quand la topologie de l'espace est complexe et/ou sa dimensionnalité élevée. (3) Nous proposons trois versions parallèles des algorithmes de type RRT pour des architectures à mémoire distribuée, afin d'améliorer la planification faisable, en espace de coût, et optimale. Nous évaluons les versions parallèles de RRT et T-RRT sur des problèmes de mouvements moléculaires, et analysons les facteurs influençant leurs performances.

A.1 Introduction

Planifier un chemin pour un système mobile, tel qu'un robot, dans un environnement complexe (typiquement, le monde réel) est un problème crucial en robotique. Résoudre ce problème a des répercussions qui peuvent aller bien au-delà du champ de la robotique. Nous pouvons commencer par mentionner les domaines applicatifs qui bénéficient directement des développements technologiques de la robotique, tels que les industries aérospatiale et manufacturière. D'autres domaines applicatifs qui sont moins directement connectés à la robotique peuvent aussi bénéficier de ces avancées : c'est le cas, par exemple, du prototypage virtuel utilisant des logiciels CAD/CAM. En effet, les tests de (dés)assemblage réalisés dans ce domaine peuvent être modélisés par des tâches de planification de chemin. L'animation graphique est un autre exemple : si les personnages virtuels sont modélisés comme des robots, les techniques de planification de chemin deviennent des outils d'animation graphique. Il existe aussi des applications médicales : par exemple, trouver un chemin minimalement-invasif pour un outil chirurgical, étant donné un modèle en trois dimensions du corps d'un patient, peut être vu comme un problème de planification de chemin. Finalement, grâce aux similitudes existantes entre les modèles structurels des robots et des molécules, les techniques de planification de chemin peuvent être appliquées en biologie structurale computationnelle.

Le problème de planification de chemin avait originellement été formulé pour un robot à éléments rigides devant se déplacer dans un environnement contenant des obstacles immobiles tout en évitant de rentrer en collision. Cette formulation est généralement nommée le "problème du déménageur de piano". Une formulation géométrique du problème de planification de chemin a été dérivée de la définition de l'espace des configurations, qui n'est autre que l'espace contenant toutes les configurations possibles du système mobile. Sur la base de ce concept, les premières méthodes proposées pour résoudre le problème de planification de chemin ont été des méthodes déterministes fournissant une solution exacte. Cependant, ces méthodes ne peuvent pas résoudre les problèmes difficiles, comme ceux mentionnés ci-dessus. La difficulté d'un problème de planification de chemin provient de la complexité du système mobile, qui est principalement exprimée par le nombre de ses degrés de liberté, ainsi que de la complexité de possibles contraintes additionnelles. Les problèmes de planification de chemin impliquant des systèmes mobiles caractérisés par de nombreux degrés de liberté sont généralement appelés "problèmes en hautes dimensions".

A.1.1 La Planification de Chemin par Échantillonnage

Contrairement aux méthodes déterministes, les méthodes de planification de chemin dites "probabilistes" ont eu beaucoup de succès du fait qu'elles ont permis de résoudre de façon efficace des problèmes complexes en hautes dimensions. Leur principe sous-jacent est d'explorer l'espace des configurations du système mobile en l'échantillonnant, et de construire un graphe représentant sa connectivité. Malgré leur simplicité conceptuelle, les planificateurs de chemin par échantillonnage se sont avérés être d'une grande valeur dans une large palette d'applications, telles que celles mentionnées précédemment. De ce fait, elles ont bénéficié d'efforts de recherche considérables durant ces 15 dernières années. Plusieurs méthodes ont été proposées, et ont ensuite été améliorées pour résoudre des problèmes difficiles, tels que la planification kinodynamique, les contraintes de chaînes cinématiques fermées, ou les environnements dynamiques. Parmi ces méthodes, l'algorithme nommé "Rapidly-exploring Random Tree" (RRT) est devenu très populaire.

Les planificateurs de chemin par échantillonnage tels que RRT ont traditionnellement été utilisés pour trouver des chemins faisables (c'est-à-dire des chemins sans collision) sans considérer la qualité de ces chemins. Ce paradigme peut être nommé la "planification de chemin faisable" (cf. Fig. A.1). Cependant, dans de nombreux domaines applicatifs il est important de générer des chemins de bonne qualité, par rapport à un critère de qualité donné. His-

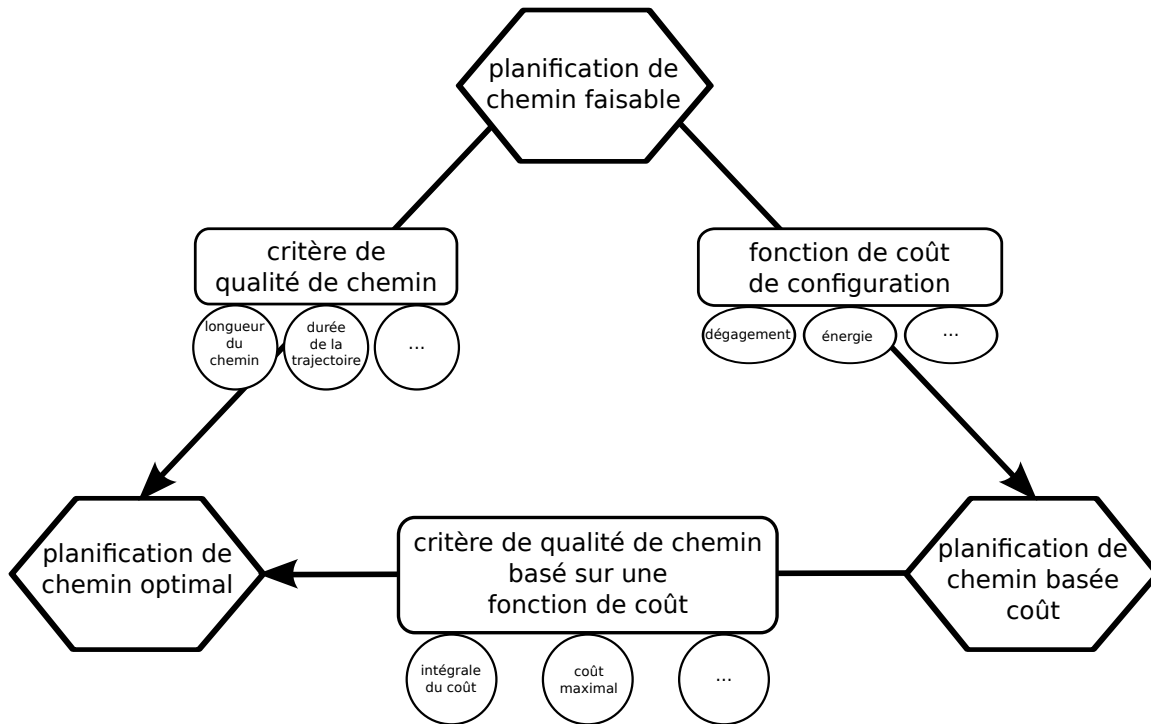


Figure A.1: Représentation schématique des relations qui existent entre les trois paradigmes de planification de chemin étudiés dans cette thèse.

toriquement, les premiers critères de qualité à avoir été utilisés sont la longueur du chemin et la durée de la trajectoire, principalement pour répondre au fait que les chemins produits par les algorithmes de planification par échantillonnage étaient généralement “saccadés”. Plus tard, la notion de “dégagement” a été introduite : l’idée sous-jacente est de générer des chemins le long desquels le système mobile reste le plus loin possible des obstacles, afin d’assurer sa propre sécurité. Quand l’utilisation d’un tel critère de qualité est importante, après qu’un chemin-solution soit calculé par le planificateur de chemin par échantillonnage, il est d’usage courant d’essayer d’améliorer la qualité de ce chemin durant une phase de post-traitement impliquant des méthodes dites de “lissage”. Néanmoins, ces méthodes ne permettent d’améliorer le chemin que localement. Nous expliquerons, plus loin, comment de meilleurs résultats peuvent être obtenus en prenant en compte le critère de qualité pendant l’exploration de l’espace des configurations.

Comme mentionné précédemment, il peut être utile de générer des chemins de bonne qualité, sur la base de l’utilisation d’un critère de qualité spécifique. Dans certains contextes applicatifs, au lieu de considérer un critère, tel que la longueur du chemin, qui évalue la qualité d’un chemin globalement, il peut être plus intéressant de s’assurer que toutes les configurations le long du chemin sont de bas coût, par rapport à une fonction de coût donnée. Par exemple, si l’on souhaite obtenir des chemins à haut dégagement, le coût d’une configuration peut être basé sur l’inverse de la distance entre le système mobile et l’obstacle le plus proche. En biologie structurale, le coût d’une conformation d’une molécule est l’énergie moléculaire. Quand une telle fonction de coût est définie sur l’espace des configurations, on appelle ce dernier un “espace de coût”. Ce paradigme peut donc être nommé la “planification de chemin en espace de coût” ou la “planification de chemin basée-coût” (cf. Fig. A.1). Ces dernières années, des variantes des algorithmes par échantillonnage classiques ont été développées afin de prendre en compte des fonctions de coût durant l’exploration de l’espace des configurations. Une de ces méthodes est une variante de RRT appelée le “Transition-based RRT” (T-RRT).

Allant au-delà de la production d'un chemin de bonne qualité, on peut rechercher un chemin optimal, par rapport à un critère de qualité de chemin donné. Ce paradigme s'appelle la "planification de chemin optimal" (cf. Fig. A.1). Appliquées à ce problème, les méthodes à base de grille classiques, telles que A^* et D^* , peuvent calculer des chemin-solutions optimaux en fonction de la résolution de la grille. Cependant, ces méthodes sont limitées aux problèmes faisant intervenir des espaces à faibles dimensions qui peuvent être discrétisés sans conduire à une explosion combinatoire. Alternativement, certains planificateurs de chemin déterministes calculent implicitement un chemin optimal par rapport à un critère spécifique. Par exemple, le *diagramme de visibilité* produit le chemin le plus court, et le *diagramme de Voronoi* génère le chemin avec dégagement optimal. Néanmoins, ces méthodes sont également limitées aux espaces à faibles dimensions et peuvent traiter uniquement les obstacles polygonaux. D'un autre côté, les planificateurs de chemin par échantillonnage classiques sont capables de traiter des espaces en hautes dimensions, mais ils produisent généralement des solutions sous-optimales. En complément, les méthodes de lissage mentionnées précédemment peuvent être utilisées pour améliorer la qualité des chemins dans une phase de post-traitement, mais elle n'offrent aucune garantie de convergence vers l'optimum global. Le premier planificateur de type RRT à fournir une telle garantie a été RRT*.

Malgré les brillants succès des planificateurs de chemin par échantillonnage classiques, leurs équivalents basés-coût n'ont été proposés que très récemment et souffrent encore de certaines limitations. En effet, de nombreux domaines applicatifs produisent des problèmes en hautes dimensions de difficulté toujours croissante, que les méthodes existantes ne peuvent pas traiter, ou seulement avec difficulté. En outre, la planification de chemin optimal est d'autant plus difficile sur de tels problèmes, car cela revient à résoudre un problème d'optimisation globale, non-linéaire, non-convexe, en hautes dimensions. De plus, les critères de qualité de chemin les plus utilisés sont toujours la longueur du chemin et la durée de la trajectoire, et les fonctions de coût de configuration les plus utilisées sont des fonctions discrètes à gros grains. Notre travail vise à traiter des fonctions de coût de configuration continues, ainsi que des critères de qualité de chemin basés sur ces fonctions de coût (cf. Fig. A.1), ce qui représente un défi beaucoup plus important. Nos objectifs sont d'améliorer les capacités et les performances des algorithmes d'échantillonnage, principalement dans le contexte de la planification en espace de coût et de la planification de chemin optimal. Nous souhaitons également étudier des applications nécessitant la création de nouvelles méthodes de planification de chemin, dans les domaines de la robotique et de la biologie structurale.

A.1.2 Contributions Algorithmiques de la Thèse

Dans cette thèse, nous proposons des extensions des algorithmes de planification de chemin par échantillonnage afin de résoudre efficacement des problèmes toujours plus difficiles faisant intervenir de complexes fonctions de coût de configuration continues. Les méthodes que nous développons sont basées sur RRT, qui peut résoudre le problème de planification de chemin faisable, ainsi que sur certaines de ses variantes : T-RRT, qui peut résoudre le problème de planification de chemin en espace de coût, et RRT*, qui peut résoudre le problème de planification de chemin optimal. Les algorithmes RRT, T-RRT et RRT* sont rapidement présentés dans le Chapitre 2. Ensuite, nous présentons les différentes contributions algorithmiques proposées dans cette thèse (cf. Fig. A.2).

Premièrement, afin d'aboutir à une planification de chemin basée-coût plus efficace, nous développons plusieurs extensions de T-RRT, en commençant par améliorer sa variante mono-directionnelle originelle, puis en proposant une variante bidirectionnelle que nous généralisons, pour finir, en une variante multi-arbres (cf. Fig. A.2 et Chapitre 3). Plus précisément, nous suggérons d'améliorer les performances du T-RRT mono-directionnel (qui avait été originellement proposé comme une extension du RRT *Extend* de base) en modifiant l'implémentation de son test de transition (cf. Section 3.1). Nous montrons également qu'utiliser le T-RRT

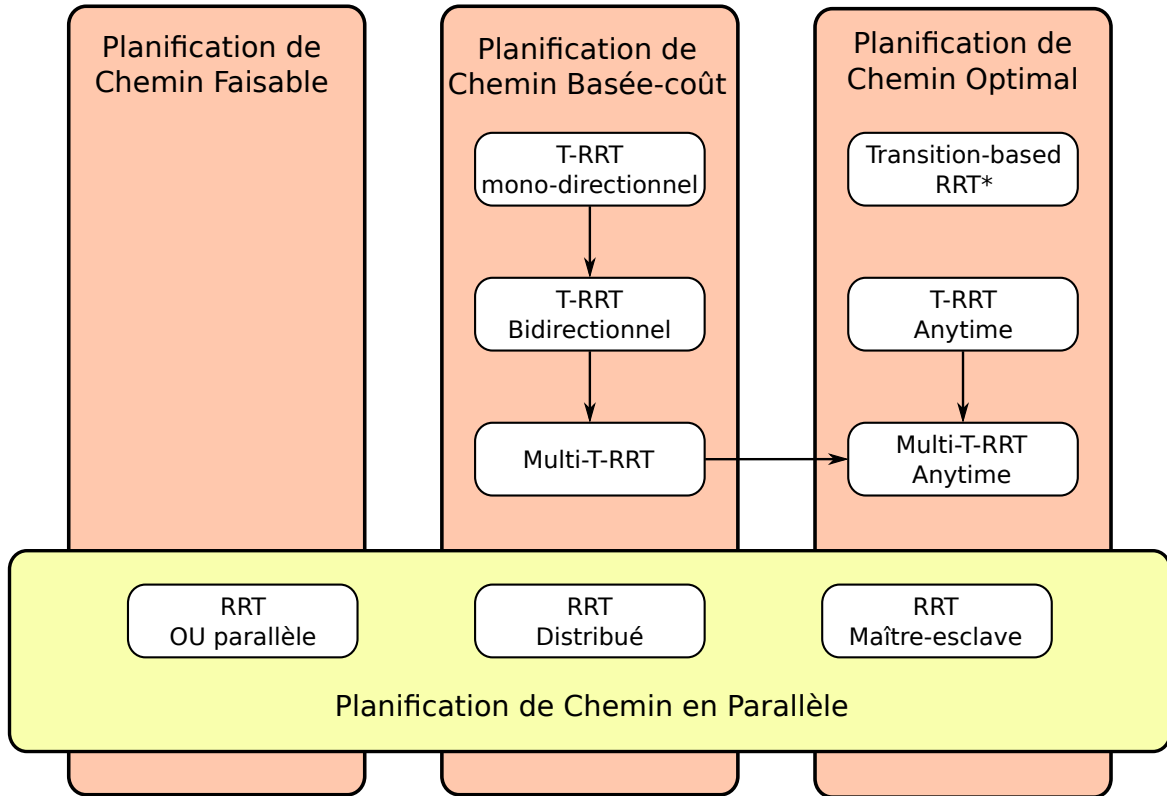


Figure A.2: Organisation schématique des principaux algorithmes proposés dans cette thèse et des relations entre eux.

Connect ou le T-RRT *Biaisé-but* produit généralement des améliorations (cf. Section 3.1). Ensuite, nous présentons une extension bidirectionnelle de T-RRT qui permet de réduire les temps de calcul et qui augmente parfois (ou sinon maintient) la qualité du chemin-solution (cf. Section 3.2). En généralisant cette approche, nous développons une extension multi-arbres de T-RRT qui peut produire un chemin passant par un ensemble de points de passage, et nous montrons que cet algorithme surpasse des planificateurs de chemin utilisant le T-RRT Bidirectionnel (cf. Section 3.3). Dans tout le Chapitre 3, nous appliquons ces extensions de T-RRT à des problèmes de robotique utiles en pratique (bien qu'uniquement simulés dans nos expérimentations) tels que des tâches d'inspection industrielle faisant intervenir des robots aériens.

Deuxièmement, pour améliorer l'efficacité des approches de planification de chemin optimal, nous combinons les concepts bénéfiques sous-jacents à T-RRT et RRT* de deux façons différentes (cf. Chapitre 4). Nous proposons une extension de RRT* nommée *Transition-based RRT** (T-RRT*) et une extension de T-RRT nommée T-RRT *Anytime* (AT-RRT). De part leurs définitions, T-RRT* et AT-RRT incluent tous deux 1) les propriétés de filtrage basées-coût du test de transition de T-RRT, favorisant la création de nouveaux nœuds dans les régions de bas coût de l'espace des configurations, et 2) la gestion des arêtes basée-qualité de RRT*, permettant à la qualité du chemin-solution de s'améliorer au fil du temps (cf. Section 4.1). Nous montrons que ces deux algorithmes sont probabilistiquement complets et asymptotiquement optimaux (cf. Section 4.2). Ensuite, nous évaluons T-RRT* et AT-RRT sur plusieurs problèmes de planification de chemin optimal, et nous montrons qu'ils convergent vers le chemin-solution optimal plus rapidement que RRT* (cf. Section 4.3). Nous montrons également que l'amélioration de performance observée est particulièrement significative quand la topologie de l'espace des configurations est complexe et/ou quand sa dimensionnalité est élevée.

Troisièmement, afin d'améliorer l'efficacité de la planification de chemin faisable, basée-côût et optimale, nous proposons trois stratégies de parallélisation pour les algorithmes de type RRT (cf. Fig. A.2 et Chapitre 5). Nous nous concentrons sur la question de paralléliser RRT sur des architectures à mémoire distribuée, ce qui nécessite d'utiliser la "Message Passing Interface" (MPI). Plus précisément, nous développons trois versions parallèles de RRT basées sur des schémas de parallélisation classiques : RRT *OU parallèle*, RRT *Distribué* et RRT *Maître-esclave* (cf. Section 5.1). Ensuite, nous évaluons les versions parallèles de RRT et T-RRT sur plusieurs problèmes de planification de chemin (cf. Section 5.2), et nous analysons les différents facteurs influençant leurs performances (cf. Section 5.3). Nos résultats expérimentaux montrent que paralléliser les algorithmes de type RRT avec MPI peut aboutir à des améliorations de performance substantielles dans plusieurs cas correspondants à de nombreux problèmes de robotique complexes et à tous les problèmes de biologie structurale. De plus, nous expliquons comment les algorithmes de type RRT introduits dans cette thèse peuvent être parallélisés (cf. Section 5.4).

Finalement, nous combinons plusieurs des approches présentées dans cette thèse pour développer de nouveaux algorithmes pouvant résoudre de nouveaux types de problèmes de planification de chemin difficiles. La raison qui nous a poussée à présenter ces méthodes séparément dans les Chapitres 3 à 5 est d'avoir la possibilité de les évaluer et de les analyser indépendamment les unes des autres. D'un autre côté, de nouvelles applications intéressantes peuvent être traitées en combinant certaines de ces méthodes. Par exemple, nous intégrons ensemble les paradigmes "anytime" et "multi-arbres" pour créer un *Multi-T-RRT Anytime* (cf. Fig. A.2) que nous utilisons en robotique (cf. Chapitre 6) et en biologie structurale (cf. Chapitre 7).

A.1.3 Applications

En complément des diverses contributions algorithmiques mentionnées ci-dessus, cette thèse contient également plusieurs contributions applicatives. En effet, nous avons appliqué plusieurs des techniques de planification de chemin par échantillonnage présentées dans cette thèse à des problèmes de planification de chemin difficiles (et parfois nouveaux) dans les domaines de la robotique (cf. Chapitre 6) et de la biologie structurale computationnelle (cf. Chapitre 7).

Robotique

En robotique, il est nécessaire de développer des méthodes pouvant prendre en compte des robots se déplaçant dans des espaces de travail de grande échelle, tels que des installations industrielles (plate-formes pétrolières, centrales électriques, aciéries, etc). Au-delà de la génération d'un chemin allant d'un point A à un point B (ce que nous appelons le problème "départ-à-but"), les tâches d'inspection industrielle nécessitent de produire un chemin passant de manière efficace par un ensemble de points de passage donnés, ce que nous appelons le problème "d'ordonnement et de recherche de chemin" (cf. Section 6.2). Pour résoudre ce problème hybride de planification de tâche et de chemin, nous proposons une variante de T-RRT appelée *Multi-T-RRT Anytime*, basée sur la combinaison de deux extensions de T-RRT présentées dans cette thèse : le *Multi-T-RRT* et le *T-RRT Anytime*. Grâce au *Multi-T-RRT Anytime*, les problèmes "d'ordonnement et de recherche de chemin" peuvent être résolus de façon purement géométrique, sans avoir à utiliser un planificateur de tâches symbolique. Nous démontrons l'intérêt de cette méthode sur un problème d'inspection industrielle faisant intervenir un robot aérien.

Un autre problème difficile que nous abordons ici est celui de la manipulation de précision en six dimensions effectuée par un système à câbles faisant intervenir plusieurs robots aériens coopérants (cf. Section 6.1). Pour cela, nous proposons un système, que nous avons appelé le *FlyCrane*, et qui est constitué d'une plate-forme attachée à trois robots volants, au moyen de

six câbles de longueurs fixes. De plus, l'élément principal de notre approche est une fonction de coût de configuration spécifique à cette application, qui prend en compte les contraintes associées à ce système robotique. Plus précisément, cette fonction de coût est basée sur des contraintes de faisabilité de tension (dérivées de l'analyse statique des manipulateurs à câbles) ainsi que sur des contraintes de poussée. Afin de valider notre approche, nous étudions deux problèmes de manipulation en simulation. Nos expérimentations montrent la supériorité de la planification de chemin basée-coût sur la planification de chemin faisable pour de tels systèmes robotiques complexes.

La plupart des problèmes de planification de chemin que nous traitons en robotique sont issus du projet européen ARCAS¹. Les objectifs de ce projet sont de développer des systèmes robotiques faisant intervenir des robots aériens coopérants, pour l'installation, l'inspection et la maintenance d'installations industrielles, dans des endroits difficiles d'accès pour des humains. Un exemple d'application possible est la construction de plate-formes d'atterrissage en terrain irrégulier, pour des avions ou des drones. Un autre exemple est l'assemblage de structures temporaires pour l'évacuation de personnes lors d'opérations de secours. Il est à noter que, malgré l'intérêt de traiter de tels problèmes dans des situations réelles, le rôle de notre équipe a été de développer et d'évaluer de nouvelles méthodes de planification de chemin dans des environnements virtuels. Les expérimentations dans le monde réel font partie des travaux en cours au sein du projet ARCAS.

Biologie Structurale Computationnelle

En biologie structurale computationnelle, tous les problèmes sont difficiles par nature, du fait de leur dimensionnalité élevée, même quand seulement de petites molécules sont considérées.

Le premier thème que nous abordons dans cette thèse est l'exploration des paysages énergétiques de peptides petits mais hautement flexibles (cf. Section 7.1). Pour cela, nous combinons deux méthodes d'échantillonnage complémentaires. 1) Nous proposons une version simplifiée de l'algorithme *Basin Hopping*, qui permet de très rapidement isoler les états structurels méta-stables d'un peptide. En biologie structurale computationnelle, le *Basin Hopping* est une méthode classique permettant d'échantillonner les minima locaux du paysage énergétique d'une molécule. 2) Ensuite, nous utilisons le *Multi-T-RRT* et le *Multi-T-RRT Anytime* pour rapidement déterminer les ensembles d'états de transition et les ensembles de chemins de transition, ainsi que les probabilités de transition, entre ces états méta-stables. Nous validons cette approche combinée sur l'alanine aux terminaisons bloquées.

Le second thème que nous abordons ici est la simulation du processus de séparation d'un complexe protéine-ligand (cf. Section 7.2). Nous proposons une approche basée sur une représentation mécanique du système moléculaire et qui, pour le moment, ne considère qu'une flexibilité partielle. La version actuelle de notre approche est purement géométrique, ce qui signifie qu'aucun calcul d'énergie moléculaire n'est effectué, et que les chemins sont validés uniquement sur la base de la détection de collisions. Cette simplification du problème nous permet d'utiliser une variante de RRT appelée *Manhattan-like RRT* (ML-RRT), dont l'efficacité exploratoire conduit à des temps de calcul très courts. Nous nous étions imposé cette obligation afin de pouvoir développer cette méthode comme une application web performante. Cet outil produit des chemins de séparation de ligand qui, en tant que première approximation, peuvent être une source d'informations utiles concernant les interactions protéine-ligand. Nous démontrons l'intérêt de cette approche sur le complexe hexamère insuline-phénol. Enfin, il est à noter que l'intégration des calculs d'énergie moléculaire dans cette approche fait partie de nos travaux en cours.

Notre travail en biologie structurale computationnelle a été effectué au sein de deux projets de recherche nommés GlucoDesign et ProtiCAD. L'objectif du projet GlucoDesign était la

¹<http://www.arcas-project.eu>

conception assistée-par-ordinateur d'outils de glycosylation enzymatique pour la synthèse de vaccins contre la shigellose endémique (ou dysenterie bacillaire). Les objectifs du projet ProtiCAD² sont de produire une méthodologie générale pour la conception de protéines, et de développer des outils informatiques pour la synthèse de nouvelles protéines.

A.2 Méthodes Apparentées de Planification de Chemin par Échantillonnage

Les algorithmes de planification de chemin par échantillonnage ont eu beaucoup de succès dans le cadre de la résolution efficace de problèmes de planification difficiles faisant intervenir des systèmes mobiles caractérisés par de nombreux degrés de liberté, autrement dit des problèmes en hautes dimensions [33,104]. Ils se sont avérés très utiles pour un large éventail de domaines applicatifs, tels que la robotique, l'industrie aérospatiale, l'industrie manufacturière, le prototypage virtuel, l'animation graphique, la médecine, et la biologie structurale computationnelle. En conséquence, ils ont bénéficié de considérables efforts de recherche durant ces quinze dernières années. Plusieurs approches ont été proposées (cf., par exemple, [75,95,106]) et ont ensuite été étendues pour traiter des problèmes difficiles, tels que la planification kinodynamique [40,75,105], les contraintes de fermeture de chaînes cinématiques [38,160], ou les environnements dynamiques [59,61,67,152,166].

Dans ce chapitre, nous passons rapidement en revue certaines de ces méthodes, dans le contexte de la planification de chemin faisable, basée-coût, et optimale. Nous insistons plus particulièrement sur les algorithmes sur lesquels nous avons travaillé dans cette thèse, ou que nous avons utilisé comme point de comparaison dans nos évaluations expérimentales. Nous passons également en revue les travaux apparentés dans le contexte de la planification de chemin en parallèle.

A.2.1 Planification de Chemin Faisable

Traditionnellement, la planification de chemin s'est focalisée sur la génération de chemins faisables, pour un système mobile, dans un environnement contenant des obstacles [102]. De façon informelle, un chemin est dit faisable s'il évite les collisions avec les obstacles et les collisions entre différentes parties articulées du système mobile (ou auto-collisions).

A.2.2 Planification de Chemin en Espace de Coût

Au lieu de construire des chemins qui sont uniquement faisables, il peut s'avérer important de générer des chemins de "bonne qualité" par rapport à un critère de qualité donné. Historiquement, comme les chemins produits par les algorithmes de planification par échantillonnage était généralement "saccadés", les premiers critères de qualité à avoir été considérés ont été la longueur du chemin et la durée de la trajectoire, évaluant ainsi la qualité d'un chemin dans sa globalité [69]. Cependant, il peut être plus intéressant de s'assurer que toutes les configurations le long d'un chemin sont de coût faible par rapport à une fonction de coût donnée. Quand une telle fonction de coût est définie sur l'espace des configurations, on appelle ce dernier un "espace de coût", et l'on parle de "planification de chemin en espace de coût" ou de "planification de chemin basée-coût".

Les premiers travaux en planification de chemin en espace de coût faisaient intervenir uniquement des fonctions de coût discrètes à gros grain [60,93]. Notre travail se concentre sur des fonctions de coût continues, ce qui est plus difficile. Par exemple, dans des problèmes de navigation en extérieur, le coût d'une configuration peut être l'élévation de la position d'un robot dans un terrain en deux dimensions. Dans les problèmes où l'on souhaite obtenir

²<http://projects.laas.fr/ProtiCAD>

des chemins à haut dégagement, le coût d'une configuration peut être basé sur l'inverse de la distance entre le système mobile et l'obstacle le plus proche [49, 83]. Des fonctions de coût bien plus complexes peuvent intervenir dans des problèmes robotiques [12, 116] ou des problèmes de biologie structurale [82].

Soit $c : \mathcal{C} \rightarrow \mathbb{R}_+$ une fonction de coût associant à chaque configuration un coût positif. Résoudre le problème de planification de chemin en espace de coût revient à résoudre le problème de planification de chemin faisable en prenant en compte la fonction de coût c lors de l'exploration de l'espace des configurations. Ceci est équivalent à effectuer un échantillonnage par rejet, en filtrant les configurations sur la base de leurs coûts. Plus précisément, chaque méthode visant à résoudre le problème de planification de chemin en espace de coût impose une contrainte de coût spécifique, évaluant chaque configuration sur la base de son coût seul ou de la variation de coût associée au mouvement local entre deux configurations.

Les premières approches traitant le problème de planification de chemin en espace de coût étaient dérivées de RRT. Malheureusement, toutes ces approches étaient focalisées sur des applications spécifiques dans le domaine de la navigation robotique en deux dimensions [57, 58, 60, 61, 107, 151], et certaines d'entre elles étaient évaluées uniquement sur des espaces de configurations faisant intervenir des fonctions de coût discrètes à gros grains [60, 61, 151]. En outre, toutes ces méthodes souffrent de divers désavantages pratiques [83]. Par exemple, certaines se basent sur l'estimation du "coût-au-but", qui tend à biaiser la recherche droit vers le but au détriment de chemins de meilleure qualité [60, 61, 151]. De plus, la méthode à seuil présentée dans [57, 58] souffre de la nature non-décroissante de son seuil et de sa haute sensibilité au taux d'accroissement du seuil [83]. Dans cette section, nous présentons une variante de RRT appelée Transition-based RRT (T-RRT), qui a eu plus de succès que les méthodes précédentes dans le cadre de la planification de chemin en espace de coût.

A.2.3 Planification de Chemin Optimal

Dans certains contextes applicatifs, au-delà de la génération de chemins de bonne qualité, il peut s'avérer intéressant de produire un chemin optimal, par rapport à un critère de qualité donné. Ce paradigme est appelé la "planification de chemin optimal" [93]. Cela peut signifier, par exemple, trouver le chemin le plus court, comme ceci est souvent le cas en robotique. Cependant, dans cette thèse, nous souhaitons ne pas nous restreindre à de tels critères, qui évaluent la qualité d'un chemin dans sa globalité en ignorant les coûts des configurations le long de ce chemin. Notre objectif est d'introduire une formulation plus générale du problème de planification de chemin optimal, où le critère de qualité de chemin est défini sur la base d'une fonction de coût de configuration caractérisant l'espace de coût. Par exemple, en robotique, ce type de planification de chemin optimal peut correspondre à la recherche d'un chemin maximisant la sécurité ; en biologie, cela signifie trouver un mouvement minimisant la variation d'énergie d'une molécule.

Appliquées au problème de planification de chemin optimal, les méthodes de grille classiques, telles que A* et D*, peuvent calculer un chemin-solution optimal en résolution [148]. Cependant, ces méthodes sont limitées à des problèmes faisant intervenir des espaces à faibles dimensions qui peuvent être discrétisés sans conduire à une explosion combinatoire. Alternativement, certains planificateurs de chemin déterministes calculent un chemin optimal par rapport à un critère de qualité spécifique. Par exemple, le *diagramme de visibilité* permet d'obtenir le chemin le plus court, et le *diagramme de Voronoï* permet de générer le chemin à dégagement optimal [102]. Néanmoins, ces méthodes sont aussi limitées aux espaces à faibles dimensions, et peuvent uniquement traiter des obstacles polygonaux. D'un autre côté, les planificateurs de chemin par échantillonnage classiques peuvent traiter des espaces en hautes dimensions, mais produisent généralement des solutions sous-optimales, parce qu'ils se focalisent sur la planification de chemin faisable. En complément de ces méthodes, après qu'un chemin-solution ait été généré, il est courant d'essayer d'améliorer la qualité de ce chemin

durant une phase de post-traitement faisant intervenir des méthodes dites de “lissage” [69]. Néanmoins, ces méthodes permettent seulement d’améliorer le chemin localement et n’offrent aucune garantie de convergence vers l’optimum global. Le premier planificateur de chemin par échantillonnage fournissant cette garantie était RRT* [90].

A.2.4 Planification de Chemin en Parallèle

L’idée d’améliorer les performances de la planification de chemin en utilisant le calcul parallèle a été explorée depuis déjà plusieurs dizaines d’années. Un article passe en revue les premiers travaux sur le sujet et suggère un schéma de classification pour présenter diverses approches de planification de chemin et leur méthodes parallèles associées [74]. Une tendance récente est d’exploiter la technologie multi-cœurs disponible sur de nombreux ordinateurs modernes, et qui permet d’avoir plusieurs threads travaillant en collaboration à la résolution d’un problème [40]. Une autre tendance récente consiste à utiliser des modèles à mémoire partagée sur des Graphics Processing Units (GPUs) [15, 17, 129, 130].

Dans le Chapitre 5 de cette thèse, nous étudions le problème de la parallélisation de RRT sur des architectures à mémoire distribuée, en utilisant la Message Passing Interface (MPI). A notre connaissance, il existe une seule tentative de parallélisation de RRT sur mémoire distribuée. Dans [44], la construction de l’arbre est distribuée entre plusieurs agents autonomes, en utilisant un modèle à messages. Cependant, aucune explication n’est donnée concernant la distribution des calculs, ou comment l’arbre est reconstruit à partir des parties construites par les agents.

Notre but est de développer des versions parallèles de RRT. Nous ne traitons pas la parallélisation des routines de RRT, telles que la détection de collision [15], ni la parallélisation des variantes de RRT, telles que le RRT *anytime* [127]. Dans le Chapitre 5, nous présentons trois versions parallèles de RRT, basées sur des schémas de parallélisation classiques : le RRT OU parallèle, le RRT Distribué, et le RRT Maître-esclave [48, 50]. Depuis que ce travail a été publié, deux extensions de notre RRT Distribué ont été proposées [81].

A.3 Planification de Chemin Basée-coût Efficace dans des Espaces de Coût Continu Complexes

En planification de chemin, au lieu de viser seulement à éviter les collisions, il peut être important de produire des chemins de bonne qualité. Ces dernières années, des variantes des algorithmes d’échantillonnage classiques ont été développées afin de prendre en compte des fonctions de coût lors de l’exploration de l’espace des configurations. Parmi ces méthodes, le Transition-based RRT (T-RRT) avait été créé comme une extension de RRT visant à effectuer spécifiquement de la planification de chemin en espace de coût (cf. Section 2.2).

T-RRT a été appliqué à des problèmes variés en robotique [12, 79, 83] (certains faisant même intervenir des interactions homme-robot [114]) et en biologie structurale computationnelle [79, 82]. Ces problèmes représentent de vrais défis car ils font intervenir des espaces en hautes dimensions et des fonctions de coût continues complexes. Notre objectif est de développer des extensions de T-RRT permettant d’améliorer ses performances et de résoudre des problèmes encore plus difficiles, ainsi que de nouveaux types de problèmes. Puisque T-RRT a été créé comme un algorithme mono-directionnel similaire au RRT *Extend*, il peut faire l’objet d’améliorations significatives, sur la base d’idées qui se sont avérées bénéfiques pour RRT, ainsi que de nouvelles idées.

Dans ce chapitre, nous présentons plusieurs extensions de T-RRT, en commençant par des améliorations de sa variante mono-directionnelle originelle, puis une variante bidirectionnelle, et enfin une variante multi-arbres. Premièrement, nous améliorons les performances du T-RRT mono-directionnel en modifiant l’implémentation de son test de transition. Nous

montrons également qu'utiliser un T-RRT *Connect* ou un T-RRT *Biaisé-but* peut parfois être profitable. Deuxièmement, nous présentons une extension bidirectionnelle de T-RRT qui permet de réduire les temps de calcul et parfois d'accroître (ou alors de maintenir) la qualité des chemins. Nous montrons également que le T-RRT *Bidirectionnel* peut produire des chemins de meilleure qualité que RRT* dans des espaces en hautes dimensions. Troisièmement, nous proposons une extension multi-arbres de T-RRT, qui peut produire un chemin passant par un ensemble de points de passage, et nous montrons qu'elle surpasse des planificateurs de chemin faisant intervenir le T-RRT Bidirectionnel. Ce chapitre contient aussi une discussion préliminaire sur une version anytime du *Multi-T-RRT*, qui est détaillée plus loin dans cette thèse (cf. Chapitres 6 et 7). Enfin, il est à noter que toutes les variantes de T-RRT présentées dans ce chapitre ont été implémentées et évaluées dans la plate-forme de planification de chemin *Move3D* [145].

A.3.1 Extensions de la Variante Mono-directionnelle de T-RRT

Dans cette section, nous présentons diverses extensions de la version originelle de T-RRT, qui avait été introduit comme un algorithme mono-directionnel semblable au RRT *Extend* (cf. Section 2.1). De part cette similarité, on pourrait penser que les modifications ayant amélioré le RRT mono-directionnel puissent être aussi avantageuses pour le T-RRT mono-directionnel. Nous montrons que ce n'est pas toujours le cas pour les variantes *Connect* et *Biaisé-but* de T-RRT. D'un autre côté, nous proposons des modifications du test de transition de T-RRT qui améliorent ses performances.

A.3.2 Extension Bidirectionnelle de T-RRT

Le RRT *Bidirectionnel* est plus performant que le RRT mono-directionnel [106]. De ce fait, dans cette section, nous évaluons s'il est possible de développer une version bidirectionnelle de T-RRT qui améliore ses performances. Il est à noter que cela ne signifie pas seulement réduire les temps de calcul, mais aussi accroître (ou au moins maintenir) la qualité des chemins. Nous comparons plusieurs implémentations possibles pour les composantes du T-RRT Bidirectionnel et sélectionnons les meilleures ; ceci concerne principalement la façon dont les arbres sont étendus et connectés. Ensuite, nous analysons les profils de coût de chemins produits par le T-RRT Bidirectionnel et le T-RRT *Extend*, afin de mieux comprendre pourquoi le premier surpasse parfois le second en termes de qualité de chemins. Nous comparons également ces deux variantes de T-RRT à RRT*, et nous montrons que T-RRT produit des chemins de meilleure qualité que RRT* dans les espaces en hautes dimensions. Enfin, nous présentons un problème d'inspection industrielle faisant intervenir un robot aérien, que le T-RRT Bidirectionnel peut résoudre efficacement, mais pas le T-RRT *Extend*.

A.3.3 Extension Multi-arbres de T-RRT

Dans cette section, nous proposons une variante multi-arbres de T-RRT, que nous avons appelé le *Multi-T-RRT*. Comme des approches multi-arbres pour la planification de chemin par échantillonnage ont été développées pour des applications diverses, avec différents objectifs, nous commençons par spécifier l'étendue de notre approche dans ce contexte général. En bref, notre but est de développer un planificateur à requête simple, capable de générer un chemin passant par un ensemble de points de passage donné, dans un environnement fixé. Puisqu'il existe de nombreux planificateurs de chemin multi-arbres faisant intervenir RRT, nous évaluons les stratégies existantes pour l'expansion et la connexion d'arbres, et nous sélectionnons les plus efficaces pour développer le *Multi-T-RRT* ; cela nécessite de prendre en compte les subtilités liées à la présence de contraintes de coût. Ensuite, nous évaluons le *Multi-T-RRT* et nous montrons qu'il surpasse les schémas de planification de chemin faisant

intervenir le RRT Bidirectionnel. Finalement, nous proposons une procédure d'ajout de cycles utiles conduisant à une version anytime du Multi-T-RRT, et permettant d'améliorer continuellement le chemin-solution. Nous montrons également que cette variante du Multi-T-RRT conduit à une convergence plus rapide vers l'optimum que PRM et que RRT_{obst way}. Seuls des résultats préliminaires sont présentés dans cette section ; la variante anytime de T-RRT est présentée plus en détails dans le Chapitre 4.

A.4 Planification de Chemin Optimal Efficace dans des Espaces de Coût Continu Complexes

En robotique, calculer un chemin-solution optimal (par rapport à un critère de qualité de chemin donné) pour un problème de planification de chemin, se fait, de façon classique, en utilisant des méthodes de grille telles que A* ou D* [148]. Cependant, ces méthodes sont connues pour être limitées à des problèmes faisant intervenir des espaces de recherche en faibles dimensions. Dans le contexte de la planification de chemin par échantillonnage, la génération d'un chemin-solution optimal est un sujet de recherche relativement nouveau [93]. Les premiers planificateurs de chemin par échantillonnage développés dans ce but ont été une variante de PRM appelée PRM*, et deux variantes de RRT appelées RRG et RRT*. En fait, il a été démontré que les algorithmes PRM*, RRG et RRT* fournissent des garanties d'optimalité asymptotique, et qu'ils peuvent, de ce fait, résoudre le problème de planification de chemin optimal (cf. Section 2.3). D'un autre côté, il a été montré que RRT (et, par extension, T-RRT) n'offre aucune garantie d'optimalité asymptotique [93].

Historiquement, les premiers critères de qualité à avoir été utilisés sont la longueur du chemin et la durée de la trajectoire. Dans cette thèse, nous souhaitons travailler avec des critères de qualité de chemin définis sur la base de fonctions de coût de configuration continues, ce qui est plus difficile. Dans ce contexte, RRT* n'est pas très efficace, car il ne prend pas en compte la fonction de coût de configuration lors de l'échantillonnage de l'espace de coût et de la création de nouveaux nœuds. En effet, RRT* peut prendre en compte uniquement un critère de qualité de chemin, lors de la création ou de la suppression d'arêtes. Ceci peut potentiellement expliquer le fait qu'il a été observé que RRT* converge vers le chemin-solution optimal assez lentement dans les espaces de coût en hautes dimensions (cf. Section 3.2 et [79]). D'un autre côté, grâce aux propriétés de filtrage de son test de transition de type Metropolis, l'exploration effectuée par T-RRT favorise les régions de bas coût de l'espace des configurations. En fait, T-RRT crée de nouveaux nœuds principalement dans ces zones favorables. Néanmoins, T-RRT ne peut pas prendre en compte le critère de qualité de chemin lors de la création d'arêtes, et, de ce fait, ne possède aucun mécanisme permettant d'améliorer la qualité du chemin-solution. Pour résumer, il apparaît clairement que RRT* et T-RRT sont basés sur des concepts différents (et complémentaires) qu'il est intéressant de combiner.

Dans ce chapitre, nous abordons la question de concevoir des algorithmes efficaces pour résoudre des problèmes de planification de chemin optimal difficiles et faisant intervenir des fonctions de coût de configuration continues complexes. Pour cela, nous nous basons sur les extensions de RRT mentionnées ci-dessus, à savoir T-RRT et RRT*. Nous combinons les deux concepts bénéfiques sous-jacents à ces méthodes : 1) les propriétés de filtrage du test de transition de T-RRT, favorisant la création de nouveaux nœuds dans les régions de bas coût de l'espace, et 2) la gestion basée-qualité des arêtes de RRT*, permettant à la qualité du chemin-solution d'augmenter avec le temps. Nous effectuons cela de deux façons différentes, en proposant une extension de RRT* nommée *Transition-based RRT** (T-RRT*) et une extension de T-RRT nommée T-RRT *Anytime* (AT-RRT). Ces deux algorithmes offrent des garanties d'optimalité asymptotique, c'est-à-dire qu'ils peuvent résoudre le problème de planification de chemin optimal. Ils permettent d'explorer efficacement les espaces de coût continu complexes, produisant des chemin-solutions de bonne qualité qui s'améliorent avec le

temps de façon “anytime”.

Nous évaluons T-RRT* et AT-RRT sur plusieurs problèmes de planification de chemin optimal, et nous montrons qu’ils convergent vers le chemin optimal plus rapidement que RRT*. Grâce aux propriétés de filtrage du test de transition, T-RRT* et AT-RRT peuvent résoudre efficacement des problèmes difficiles sur lesquels RRT* converge très lentement. Nous présentons plusieurs exemples illustrant divers facteurs pouvant rendre un problème de planification de chemin optimal difficile à résoudre. 1) Si le problème fait intervenir un espace de travail de grande échelle, même en faibles dimensions, favoriser les régions de bas coût évite de gaspiller du temps à explorer tout l’espace. 2) Si l’espace comprend plusieurs classes d’homotopie entre lesquelles il est difficile de passer, même en faibles dimensions, utiliser le test de transition peut biaiser la recherche vers la classe contenant le chemin optimal et éviter de se retrouver bloqué dans une classe sous-optimale. 3) Si le problème est en hautes dimensions, il est intrinsèquement complexe, car l’espace de recherche est très grand, par nature, et peut potentiellement contenir de nombreuses classes d’homotopie.

A.4.1 Algorithmes

Dans cette section, nous combinons les concepts bénéfiques sous-jacents à T-RRT et RRT*, pour développer de nouveaux algorithmes héritant de leurs points forts mais pas de leurs points faibles respectifs. Le premier algorithme, nommé *Transition-based RRT** (T-RRT*), est basé sur l’intégration du test de transition de type Metropolis de T-RRT dans RRT*. L’idée est de favoriser l’exploration des régions de bas coût de l’espace, en prenant en compte la fonction de coût de configuration (comme le fait T-RRT), tout en maintenant les propriétés asymptotiques de RRT*. Le second algorithme, nommé T-RRT *Anytime* (AT-RRT), consiste à enrichir T-RRT avec un comportement “anytime” en intégrant une procédure ajoutant des cycles utiles (par rapport au critère de qualité de chemin) dans le graphe construit sur l’espace des configurations, comme cela est fait pour PRM dans [124]. L’idée est d’obtenir rapidement une première solution de bonne qualité et, ensuite, de continuer l’exploration pour que la solution s’améliore continuellement et converge vers le chemin optimal.

A.4.2 Analyse Théorique

Dans cette section, nous montrons que T-RRT* et AT-RRT sont tous deux probabilistiquement complets et asymptotiquement optimaux.

A.4.3 Évaluation

Dans cette section, nous évaluons T-RRT* et AT-RRT sur plusieurs problèmes de planification de chemin optimal, qui diffèrent en termes de dimensionnalité, de complexité géométrique, et de type de fonction de coût de configuration.

A.5 Planification de Chemin en Parallèle sur des Architectures à Mémoire Distribuée

En dépit du niveau de succès atteint par la planification de chemin par échantillonnage, certains problèmes difficiles représentent toujours un défi trop grand pour les méthodes actuelles. Une façon de relever ce défi est de proposer des versions améliorées des approches existantes, au niveau algorithmique, comme nous l’avons fait dans les Chapitres 3 et 4. Une autre solution est d’optimiser les approches existantes, au niveau implémentation, pour les rendre plus efficaces. Dans le contexte des algorithmes de type RRT, plusieurs techniques ont été proposées, telles que la réduction de la complexité de la recherche du plus proche voisin [162], le contrôle dynamique des domaines d’échantillonnage [84], ou la réduction de la dispersion

des échantillons [109]. Au lieu d'optimiser les différentes composantes de ces algorithmes, une approche plus globale peut être adoptée, sur la base du calcul parallèle.

Dans ce chapitre, nous abordons la question d'améliorer les performances des algorithmes de type RRT en exploitant l'accélération intrinsèque au calcul parallèle. Certains résultats ont déjà été obtenus dans ce sens (cf. Section 2.4). Néanmoins, les travaux existants considèrent principalement les architectures à mémoire partagée et le parallélisme à petite échelle, jusqu'à 16 processeurs. Dans ce travail, nous nous intéressons à ce qui peut être obtenu grâce au parallélisme à grande échelle. Nous nous focalisons sur la parallélisation des algorithmes de type RRT sur des architectures à mémoire distribuée de grande échelle, ce qui nécessite d'utiliser la Message Passing Interface (MPI).

Ce travail de parallélisation peut bénéficier à la planification de chemin faisable, basée-coût, et optimale. Cependant, les exemples présentés dans ce chapitre se concentrent sur la planification de chemin basée-coût et optimale. Par souci de clarté, nous commençons par présenter des versions parallèles de la variante mono-directionnelle de base de RRT. Ensuite, nous réalisons des évaluations expérimentales faisant intervenir les algorithmes RRT et T-RRT. Pour finir, nous expliquons comment les nouveaux algorithmes de type RRT proposés dans les autres chapitres, tels que la variante multi-arbres (anytime) de T-RRT, peuvent être parallélisés (cf. Section 5.4).

Nous comparons trois versions parallèles de RRT, basées sur des schémas de parallélisation bien connus : le RRT *OU parallèle*, le RRT *Distribué* et le RRT *Maître-esclave*. En complément des algorithmes eux-mêmes, nous présentons les principales spécificités techniques impliquées dans leur développement (Section 5.1). Notre contribution se concentre sur l'évaluation de ces algorithmes sur plusieurs problèmes de planification de chemin, afin de montrer leurs différences de comportement (Section 5.2). Nous analysons également leurs performances afin de comprendre l'impact de plusieurs caractéristiques des problèmes étudiés (Section 5.3). Nos évaluations montrent que paralléliser les algorithmes de type RRT avec MPI peut apporter des améliorations de performance significatives dans deux cas : 1) les problèmes dont la variabilité du temps de calcul séquentiel est élevée peuvent bénéficier du RRT *OU parallèle* ; 2) les problèmes pour lesquels le coût d'une expansion de RRT est élevé peuvent bénéficier du RRT *Distribué* et du RRT *Maître-esclave*. Tous les problèmes de biologie structurale, ainsi que de nombreux problèmes de robotique, sont caractérisés par des expansions de RRT ayant un coût de calcul important, et peuvent donc bénéficier de ces algorithmes parallèles.

A.5.1 Parallélisation de RRT

Pour des raisons de passage à l'échelle, nous parallélisons RRT sur des architectures à mémoire distribuée, en utilisant la Message Passing Interface³ (MPI), une des approches les plus répandues en programmation parallèle. L'architecture matérielle considérée comprend p processeurs, chacun ayant son espace d'adressage respectif. Puisque ce paradigme n'impose aucune restriction sur le matériel sous-jacent et nécessite de paralléliser explicitement les algorithmes, il permet une large portabilité : un algorithme développé selon cette approche peut aussi être utilisé en mémoire partagée.

A.5.2 Expérimentations avec RRT et T-RRT

Avant de présenter nos résultats expérimentaux, nous définissons les métriques utilisées pour évaluer les algorithmes parallèles. Nous présentons également la plate-forme parallèle utilisée, et les problèmes de planification de chemin étudiés.

³<http://www.mpi-forum.org>

A.5.3 Analyse des Algorithmes Parallèles

Nous présentons une analyse détaillée de nos trois algorithmes parallèles, afin de mieux comprendre les résultats obtenus en termes d'accélération. Nous étudions l'influence du type de problème, du nombre de processeurs, et du coût d'une expansion de RRT.

A.5.4 Application à d'Autres Algorithmes de Type RRT

Dans cette section, nous expliquons brièvement comment les nouveaux algorithmes de type RRT proposés dans cette thèse peuvent être parallélisés.

A.6 Application à des Problèmes de Robotique Complexes

Dans ce chapitre, nous présentons deux applications des algorithmes étudiés dans cette thèse, dans le domaine de la robotique. Nous montrons qu'en développant des fonctions de coût de configuration continues sophistiquées, et plus adaptées à une application spécifique que la fonction de coût basée sur le dégagement (classiquement utilisée dans les exemples académiques), il est possible de traiter des problèmes réalistes complexes. En effet, de telles fonctions de coût peuvent prendre en compte les contraintes inhérentes à des systèmes robotiques complexes, tels que les systèmes aériens à câbles, impliquant plusieurs robots volants. Nous montrons également qu'en combinant plusieurs des extensions de T-RRT proposées dans cette thèse, il est possible de traiter des problèmes de planification de chemin plus difficiles que les problèmes classiques de type "départ-à-but", tels que les problèmes "d'ordonnancement et de recherche de chemin".

Ces deux types de problèmes sont caractéristiques des sujets traités dans le projet ARCAS. Un des objectifs de ce projet est de développer des systèmes robotiques pour l'assemblage, l'inspection et la maintenance d'installations industrielles difficiles d'accès pour les humains.

A.6.1 Manipulation 6-D avec un Système à Câbles Aérien

Dans cette section, nous proposons une approche de planification de chemin innovante pour la manipulation quasi-statique sûre, en six dimensions, avec un système à câbles aérien. La nouveauté de cette approche réside dans l'utilisation combinée d'un algorithme de planification de chemin en espace de coût et de résultats dérivant de l'analyse statique des manipulateurs à câble. Sur la base de contraintes de faisabilité de tension, appliquées aux câbles, et de contraintes de poussée, appliquées aux robots volants, nous caractérisons l'ensemble des configurations faisables du système. En outre, l'expression de ces contraintes conduit à un critère permettant d'évaluer la qualité d'une configuration. Ceci permet de définir une fonction de coût sur l'espaces des configurations, que nous exploitons pour calculer des chemins de bonne qualité en utilisant T-RRT. Dans le cadre de notre approche, nous proposons également un système aérien à câbles que nous avons appelé le *FlyCrane*. Il est constitué d'une plateforme attachée à trois robots volants, à l'aide de six câbles de longueurs fixes. Nous validons l'approche proposée, en simulation, sur deux problèmes de manipulation quasi-statique en six dimensions faisant intervenir un tel système, et nous montrons l'intérêt de prendre en compte la fonction de coût pour de telles tâches de planification de chemin.

A.6.2 Problèmes "d'Ordonnancement et de Recherche de Chemin"

Dans cette section, nous proposons une nouvelle variante de T-RRT appelée le *Multi-T-RRT Anytime*, basée sur la combinaison de deux extensions de T-RRT : le Multi-T-RRT (cf. Chapitre 3) et le T-RRT Anytime (cf. Chapitre 4). Cet algorithme est particulièrement utile pour résoudre des problèmes "d'ordonnancement et de recherche de chemin", c'est-à-dire pour générer un chemin de bonne qualité passant par plusieurs points de passage non

ordonnés a priori. En utilisant le Multi-T-RRT Anytime, ces problèmes peuvent être résolus de façon purement géométrique, sans avoir à utiliser un planificateur de tâche symbolique. Afin de démontrer ses capacités, nous appliquons le Multi-T-RRT Anytime à un problème d'inspection industrielle concret faisant intervenir un robot aérien.

A.7 Application à des Problèmes de Biologie Structurale

Dans ce chapitre, nous présentons deux applications des algorithmes de planification par échantillonnage, dans le domaine de la biologie structurale computationnelle. Le défi majeur de ce domaine est que tous les problèmes sont intrinsèquement difficiles, du fait de leur dimensionnalité élevée, même quand seules de petites molécules sont considérées. De nombreuses questions de recherche intéressantes peuvent être traitées en utilisant des algorithmes d'inspiration robotique, car ils permettent d'explorer efficacement l'espace des conformations d'une molécule ou même d'un complexe moléculaire. Dans cette thèse, nous nous concentrons sur deux thèmes différents : l'exploration du paysage énergétique d'un petit peptide, et la simulation du processus de séparation d'un complexe protéine-ligand.

Tous les algorithmes présentés dans ce chapitre ont été développés au sein d'un logiciel appelé MoMA (pour Molecular Motion Algorithms) qui contient un ensemble d'algorithmes d'inspiration robotique pour la simulation de mouvements moléculaires.

A.7.1 Exploration du Paysage Énergétique d'un Petit Peptide

Dans cette section, nous abordons la question de l'obtention d'une caractérisation complète du paysage énergétique d'un peptide petit mais hautement flexible. Pour cela, nous suggérons de combiner deux méthodes d'échantillonnage complémentaires : le Basin Hopping et le Multi-T-RRT (Anytime). Nous proposons une version simplifiée de l'algorithme de Basin Hopping, qui permet de révéler rapidement les états structurels méta-stables d'un peptide, ainsi que les bassins énergétiques correspondants, dans le paysage. Ensuite, nous utilisons des variantes de l'algorithme T-RRT pour déterminer rapidement les ensembles d'états de transition et les ensembles de chemins de transition, ainsi que les probabilités de transition, entre ces états méta-stables. Plus précisément, nous proposons un Multi-T-RRT Anytime spécifique aux problèmes de biologie structurale, basé sur le Multi-T-RRT (cf. Chapitre 3) et sur le T-RRT Anytime (cf. Chapitre 4). Nous démontrons l'intérêt de cette approche combinant Basin Hopping et T-RRT sur l'alanine aux terminaisons bloquées.

A.7.2 Simulation de la Séparation d'un Complexe Protéine-Ligand

Dans cette section, nous abordons la question de la simulation des interactions protéine-ligand se produisant loin du site actif de la protéine, durant la liaison ou la séparation du ligand. Dans une première étape allant vers ce à quoi nous souhaiterions aboutir, l'approche que nous présentons ici est purement géométrique. Cette approche est basée sur une représentation mécanique du système moléculaire, considérant une flexibilité partielle, et sur l'application d'une variante de RRT appelée Manhattan-like RRT (ML-RRT), pour explorer l'espace des conformations. Cela signifie qu'aucune énergie moléculaire n'est calculée et que les mouvements sont validés uniquement sur la base de l'absence de collision. Cette approche purement géométrique, couplée avec un algorithme d'exploration efficace, permet de simuler la séparation d'un ligand en un temps de calcul très court. Le fait d'aboutir à des temps de calcul très courts était une contrainte que nous nous étions imposé afin de pouvoir développer un serveur web efficace. Cet outil en ligne génère des chemins de séparation pour un ligand, qui, en tant que première approximation, peuvent fournir des informations utiles concernant les interactions protéine-ligand. Nous démontrons l'intérêt de cet outil informatique sur le complexe hexamère insuline-phénol.

A.8 Conclusion

Dans cette thèse, nous avons abordé les questions de l'amélioration de l'efficacité et de l'enrichissement des capacités des algorithmes d'échantillonnage dans le contexte de la planification de chemin faisable, basée-coût, et optimale. Notre intention était de traiter des problèmes en hautes dimensions difficiles, faisant intervenir des fonctions de coût continues complexes. De ce fait, notre travail a été plus spécifiquement orienté vers la planification de chemin basée-coût et optimale. Ceci était justifié par le fait que, bien que les méthodes d'échantillonnage classiques aient atteintes une forte reconnaissance, leurs équivalentes en espace de coût n'ont été développées que récemment et nécessitent des travaux de recherche supplémentaires. Dans les Chapitres 3 à 5, nous avons proposé plusieurs extensions de planificateurs de chemin de type RRT, basés sur les algorithmes T-RRT et RRT*. Dans les Chapitres 6 et 7, nous avons présenté des applications variées de ces approches nouvelles, dans les domaines de la robotique et de la biologie structurale computationnelle.

A.8.1 Résumé des Contributions Algorithmiques

Premièrement, nous avons proposé plusieurs extensions de l'algorithme T-RRT visant à améliorer la planification de chemin en espace de coût [49]. Plus précisément, nous avons amélioré sa variante mono-directionnelle en optimisant son test de transition. Nous avons également proposé une variante bidirectionnelle de T-RRT, prenant en compte les contraintes de coût. Nous avons montré que, dans le contexte d'un problème de planification de chemin de type "départ-à-but", utiliser le T-RRT Bidirectionnel était une meilleure stratégie qu'utiliser le T-RRT Connect ou le T-RRT Biaisé-but mono-directionnels. Ensuite, en généralisant cette approche bidirectionnelle, nous avons proposé une variante multi-arbres de T-RRT, qui peut générer un chemin passant par un ensemble de points de passage [52].

Deuxièmement, nous avons expliqué comment résoudre les problèmes de planification de chemin optimal de façon plus efficace que ce que RRT* ne permet à présent dans les espaces de coût. En combinant les concepts sous-jacents à T-RRT et RRT*, c'est-à-dire la création de nœuds basée-coût et la gestion d'arêtes basée-qualité, nous avons proposé deux nouveaux algorithmes d'échantillonnage qui sont asymptotiquement optimaux : le Transition-based RRT* et le T-RRT Anytime. Nous avons également montré qu'ils convergent plus vite que RRT* vers le chemin optimal, en particulier quand la topologie de l'espace des configurations est complexe et/ou quand sa dimensionnalité est élevée [51].

Troisièmement, nous avons présenté trois versions parallèles de RRT destinées aux architectures à mémoire distribuée de grande échelle : le RRT OU parallèle, le RRT Distribué, et le RRT Maître-esclave [48]. Nous avons également expliqué comment ces schémas de parallélisation peuvent être appliqués aux algorithmes de type RRT autres que ceux introduits dans cette thèse. Nous avons observé qu'utiliser ces algorithmes parallèles peut réduire les temps de calcul de façon significative sur des problèmes de robotique complexes ou des problèmes de biologie structurale [50].

Pour finir, nous avons suggéré que combiner certaines de ces approches nouvelles pouvait permettre de résoudre de nouveaux types de problèmes de planification. Par exemple, en intégrant le T-RRT Anytime et le Multi-T-RRT, nous avons développé le Multi-T-RRT Anytime que nous avons utilisé dans plusieurs applications.

A.8.2 Résumé des Contributions Applicatives

En robotique, nous avons tout d'abord utilisé le Multi-T-RRT Anytime pour résoudre les problèmes "d'ordonnancement et de recherche de chemin" [52]. Avec cette approche, ces problèmes peuvent être résolus de façon purement géométrique, sans avoir à utiliser de planificateur de tâche symbolique. Comme exemple, nous avons présenté, en simulation, un

problème d'inspection industrielle faisant intervenir un robot aérien. Deuxièmement, nous avons expliqué comment une approche basée-coût peut être utile pour réaliser des tâches de manipulation précise en six dimensions, avec le FlyCrane, un système à câbles composé de trois robots aériens [115, 116]. Pour cela, nous avons défini une fonction de coût de configuration spécifique à cette application et prenant en compte les contraintes de cette plate-forme robotique. Nous avons évalué cette approche, en simulation, sur des problèmes de manipulation 6-D faisant intervenir ce système.

En biologie structurale computationnelle, nous avons utilisé le Multi-T-RRT (Anytime), en complément d'une autre méthode d'échantillonnage appelée Basin Hopping, pour produire une caractérisation complète du paysage énergétique d'un petit peptide flexible [47, 53]. Cette approche combinée nous a permis de déterminer les états structuraux méta-stables du peptide, ainsi que les ensembles d'états de transition, les ensembles de chemins de transition, et les probabilités de transition, entre ces états méta-stables. Nous avons validé cette approche avec l'alanine aux terminaisons bloquées. Une autre application que nous avons présenté ici est la simulation du processus de séparation d'un complexe protéine-ligand [45, 46]. Dans une première étape, nous avons proposé une approche géométrique utilisant le Manhattan-like RRT, ce qui, du fait d'avoir des temps de calcul extrêmement faibles, nous a permis de l'implémenter comme un serveur web. Nous avons démontré l'intérêt de cette méthode sur le complexe hexamère insuline-phénol.

A.8.3 Perspectives de Travaux Futurs

Comme première direction de recherche possible pour la suite, nous pourrions commencer par améliorer les méthodes présentées dans cette thèse. En effet, certaines de nos approches peuvent être poussées plus loin pour atteindre un meilleur niveau d'efficacité. Premièrement, dans le contexte de la planification de chemin en espace de coût, les algorithmes de type T-RRT peuvent être améliorés en prenant en compte la fonction de coût de façon plus performante. En effet, un problème inhérent à la stratégie d'échantillonnage par rejet utilisée actuellement est le gâchis de ressources qu'elle engendre. Une solution possible est de guider partiellement l'exploration du paysage de coût en exploitant, par exemple, des méthodes de gradient, comme cela est fait dans [12]. Cependant, cela nécessite de maintenir un bon équilibre entre le biais résultant de cette approche et les bonnes caractéristiques exploratoires des algorithmes de type RRT. Deuxièmement, dans le contexte de la planification de chemin optimal, nous pouvons développer un planificateur de chemin plus performant, basé sur AT-RRT, T-RRT*, ou une combinaison des deux. Cela nécessite de poursuivre l'analyse de AT-RRT et RRT*, et de déterminer quelle stratégie fonctionne le mieux en général ou sur des classes de problèmes spécifiques. Enfin, dans le contexte de la planification de chemin en parallèle, un schéma de parallélisation plus performant pourrait être obtenu en combinant les trois approches parallèles présentées dans cette thèse.

Sur un autre plan, nous pourrions essayer de tirer un meilleur profit du graphe produit par les variantes anytime des méthodes de type T-RRT, car cela pourrait avoir des répercussions bénéfiques. Premièrement, du côté de la biologie structurale computationnelle, dans le cadre de l'exploration du paysage énergétique d'un peptide, nous pourrions exploiter ce graphe pour décrire les ensembles d'états de transition et les ensembles de chemins de transition, ainsi que pour estimer les probabilités de transition entre les états méta-stables. Cela aboutirait à une meilleure utilisation des ressources computationnelles que ce que n'offrent nos méthodes actuelles, et cela nous permettrait d'étudier des peptides plus gros. Deuxièmement, du côté de la robotique, au lieu de générer des chemins passant par des points de passage considérés comme ayant tous le même statut, nous pourrions envisager d'autres applications utiles. Par exemple, nous pourrions résoudre des problèmes faisant intervenir une seule configuration initiale et plusieurs configurations comme buts potentiels. Les différents chemins extraits de ce graphe pourraient représenter plusieurs façons de résoudre ce problème hybride de

planification de tâche et de chemin, nous permettant ainsi de choisir le meilleur chemin ou de modifier le chemin actuel.

En robotique, combiner les méthodes que nous avons proposé dans cette thèse offre des possibilités intéressantes. Par exemple, dans le contexte de la manipulation en six dimensions avec le FlyCrane, l'approche proposée pourrait être enrichie par l'utilisation de variantes plus sophistiquées de T-RRT. Ceci est particulièrement vrai pour le T-RRT Anytime, car nous pourrions alors produire le chemin optimal pour réaliser une tâche de manipulation donnée. En outre, le Multi-T-RRT pourrait être utilisé pour résoudre des problèmes complexes de planification de tâche et de chemin englobant la manipulation de plusieurs objets pour l'assemblage ou le désassemblage d'une grande structure.

Certains algorithmes que nous avons proposé dans cette thèse peuvent être utilisés dans d'autres contextes de planification que ceux que nous avons étudié ici. Premièrement, les variantes anytime des méthodes de type T-RRT peuvent être utiles pour la re-planification de chemin. Puisqu'elles construisent un graphe contenant des cycles, ces méthodes fournissent des chemins alternatifs qui deviennent disponibles dans le cas où le chemin-solution actuel est invalidé du fait d'erreurs dans le modèle ou de déplacements des obstacles. Deuxièmement, ces algorithmes anytime peuvent être utilisés pour la planification en ligne. Tandis que seule une partie du chemin-solution actuel est exécutée par le robot, le reste du chemin peut être davantage optimisé [94].

En biologie structurale computationnelle, de nouveaux problèmes intéressants peuvent être étudiés en utilisant les algorithmes présentés dans cette thèse. Ceci est particulièrement vrai pour la simulation d'interactions protéine-ligand, car nous n'avons pas encore appliqué les paradigmes multi-arbres et anytime à ces problèmes. Premièrement, si plusieurs conformations du complexe protéine-ligand sont disponibles, une approche multi-arbres pourrait nous permettre de générer plusieurs chemins de séparation possibles en même temps, et de déterminer lequel est le plus probable. Deuxièmement, une approche anytime pourrait nous permettre de trouver le chemin de séparation optimal par rapport à un critère de qualité de chemin donné, tel que la résistance minimale ou le flux maximal.

Le domaine de la biologie structurale produit des exemples pouvant nécessiter d'énormes quantités de ressources computationnelles. Dans le cadre de l'exploration de l'espace des conformations d'une molécule ou d'un complexe moléculaire, construire plusieurs arbres simultanément s'est déjà avéré utile pour les applications présentées dans cette thèse. Pour aller encore plus loin, au lieu de construire seulement quelques arbres dans l'espace de recherche, il peut s'avérer utile de construire des centaines, voire des milliers, d'arbres simultanément. Le paradigme de planification parallèle peut permettre aux variantes multi-arbres des méthodes de type T-RRT de relever ce défi, en entrecroisant plusieurs niveaux de parallélisation. Par exemple, nous pouvons envisager de combiner les trois niveaux suivants : 1) distribuer la construction des arbres entre plusieurs groupes de processeurs ; 2) partager la construction de chaque arbre entre plusieurs processeurs ; 3) paralléliser les composantes les plus coûteuses en calcul de l'expansion de T-RRT. L'utilisation de telles versions parallèles des algorithmes multi-arbres nous permettrait d'exploiter de nombreuses ressources computationnelles et de résoudre des problèmes extrêmement complexes.

Bibliography

- [1] Y. Abbasi-Yadkori, J. Modayil, and C. Szepesvári. Extending rapidly-exploring random trees for asymptotically optimal anytime motion planning. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 127–132, 2010.
- [2] I. Aguinaga, D. Borro, and L. Matey. Parallel RRT-based path planning for selective disassembly planning. *The International Journal of Advanced Manufacturing Technology*, 36(11-12):1221–1233, 2008.
- [3] I. Al-Bluwi, T. Siméon, and J. Cortés. Motion planning algorithms for molecular simulations: A survey. *Computer Science Review*, 6(4):125–143, 2012.
- [4] J. Albus, R. Bostelman, and N. Dagalakis. The NIST robocrane. *Journal of Robotic Systems*, 10(5):709–724, 1993.
- [5] R. Alterovitz, S. Patil, and A. Derbakova. Rapidly-exploring roadmaps: weighing exploration vs. refinement in optimal motion planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3706–3712, 2011.
- [6] N. M. Amato and L. K. Dale. Probabilistic roadmap methods are embarrassingly parallel. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 688–694, 1999.
- [7] P. Amodeo, F. Naider, D. Picone, T. Tancredi, and P. A. Temussi. Conformational sampling of bioactive conformers: A low-temperature NMR study of ¹⁵N-Leu-Enkephalin. *Journal of Peptide Science*, 4(4):253–265, 1998.
- [8] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18(6):911–922, 2002.
- [9] C. Belta, J. M. Esposito, J. Kim, and V. Kumar. Computational techniques for analysis of genetic network dynamics. *The International Journal of Robotics Research*, 24(2-3):219–235, 2005.
- [10] J. L. Bentley. Experiments on traveling salesman heuristics. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 91–99, 1990.
- [11] H. Berchtold and R. Hilgenfeld. Binding of phenol to R6 insulin hexamers. *Biopolymers*, 51(2):165–172, 1999.
- [12] D. Berenson, T. Siméon, and S. S. Srinivasa. Addressing cost-space chasms in manipulation planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4561–4568, 2011.
- [13] D. Berenson, S. Srinivasa, and J. J. Kuffner. Task space regions: a framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460, 2011.

-
- [14] M. Bernard, K. Kondak, I. Maza, and A. Ollero. Autonomous transportation and deployment with aerial robots for search and rescue missions. *Journal of Field Robotics*, 28(6):914–931, 2011.
- [15] J. Bialkowski, S. Karaman, and E. Frazzoli. Massively parallelizing the RRT and the RRT*. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3513–3518, 2011.
- [16] L. Biedermannová, Z. Prokop, A. Gora, E. Chovancová, M. Kovács, J. Damborský, and R. C. Wade. A single mutation in a tunnel to the active site changes the mechanism and kinetics of product release in haloalkane dehalogenase LinB. *The Journal of Biological Chemistry*, 287(34):29062–29074, 2012.
- [17] A. Bleiweiss. Parallel compact roadmap construction of 3D virtual environments on the GPU. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5007–5013, 2010.
- [18] O. Bohigas, M. Manubens, and L. Ros. Navigating the wrench-feasible C-space of cable-driven hexapods. In T. Bruckmann and A. Pott, editors, *Cable-Driven Parallel Robots*, volume 12 of *Mechanisms and Machine Science*, pages 53–68. Springer Berlin Heidelberg, 2013.
- [19] P. G. Bolhuis, C. Dellago, and D. Chandler. Reaction coordinates of biomolecular isomerization. *Proceedings of the National Academy of Sciences of the United States of America*, 97(11):5877–5882, 2000.
- [20] K. W. Borrelli, A. Vitalis, R. Alcantara, and V. Guallar. PELE: Protein energy landscape exploration. A novel Monte Carlo based technique. *Journal of Chemical Theory and Computation*, 1(6):1304–1311, 2005.
- [21] P. Bosscher, A. T. Riechel, and I. Ebert-Uphoff. Wrench-feasible workspace generation for cable-driven robots. *IEEE Transactions on Robotics*, 22(5):890–902, 2006.
- [22] K. Bouyarmane and A. Kheddar. Static multi-contact inverse problem for multiple humanoid robots and manipulated objects. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 8–13, 2010.
- [23] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan. RRTs for nonlinear, discrete, and hybrid planning and control. In *Proc. IEEE Conference on Decision and Control (CDC)*, pages 657–663, 2003.
- [24] S. Cambon, R. Alami, and F. Gravot. A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126, 2009.
- [25] S. Carpin and E. Pagello. On parallel RRTs for multi-robot systems. In *Proc. International Conference of the Italian Association for Artificial Intelligence (AI*IA)*, pages 834–841, 2002.
- [26] S. Caselli and M. Reggiani. ERPP: An experience-based randomized path planner. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1002–1008, 2000.
- [27] D. Challou, D. Boley, M. Gini, V. Kumar, and C. Olson. Parallel search algorithms for robot motion planning. In K. Gupta and A. P. del Pobil, editors, *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, pages 115–131. John Wiley & Sons Ltd., 1998.

- [28] R. Chaloupková, J. Sýkorová, Z. Prokop, A. Jesenská, M. Monincová, M. Pavlová, M. Tsuda, Y. Nagata, and J. Damborský. Modification of activity and specificity of haloalkane dehalogenase from *Sphingomonas paucimobilis* UT26 by engineering of its entrance tunnel. *The Journal of Biological Chemistry*, 278(29):52622–52628, 2003.
- [29] P. Cheng, E. Frazzoli, and S. M. LaValle. Improving the performance of sampling-based motion planning with symmetry-based gap reduction. *IEEE Transactions on Robotics*, 24(2):488–494, 2008.
- [30] P. Cheng and S. M. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 43–48, 2001.
- [31] P. Cheng and S. M. LaValle. Resolution complete Rapidly-exploring Random Trees. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 267–272, 2002.
- [32] J. D. Chodera, W. C. Swope, J. W. Pitera, and K. A. Dill. Long-time protein folding dynamics from short-time molecular dynamics simulations. *Multiscale Modeling and Simulation*, 5(4):1214–1226, 2006.
- [33] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.
- [34] M. Clifton, G. Paul, N. Kwok, and D. Liu. Evaluating performance of multiple RRTs. In *Proc. IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 564–569, 2008.
- [35] J. Cortés, L. Jaillet, and T. Siméon. Molecular disassembly with RRT-like algorithms. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3301–3306, 2007.
- [36] J. Cortés, L. Jaillet, and T. Siméon. Disassembly path planning for complex articulated objects. *IEEE Transactions on Robotics*, 24(2):475–481, 2008.
- [37] J. Cortés, D. Le, R. Iehl, and T. Siméon. Simulating ligand-induced conformational changes in proteins using a mechanical disassembly method. *Physical Chemistry Chemical Physics*, 12(29):8268–8276, 2010.
- [38] J. Cortés and T. Siméon. Sampling-based motion planning under kinematic loop-closure constraints. In M. Erdmann, D. Hsu, M. Overmars, and A. F. van der Stappen, editors, *Algorithmic Foundations of Robotics VI*, pages 75–90. Springer-Verlag, 2005.
- [39] J. Cortés, T. Siméon, V. Ruiz de Angulo, D. Guieysse, M. Remaud-Siméon, and V. Tran. A path planning approach for computing large-amplitude motions of flexible molecules. *Bioinformatics*, 21(Suppl. 1):i116–i125, 2005.
- [40] I. A. Şucan and L. E. Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. In G. S. Chirikjian, H. Choset, M. Morales, and T. Murphey, editors, *Algorithmic Foundations of Robotics VIII*, pages 449–464. Springer-Verlag, 2010.
- [41] S. Dalibard and J.-P. Laumond. Control of probabilistic diffusion in motion planning. In G. S. Chirikjian, H. Choset, M. Morales, and T. Murphey, editors, *Algorithmic Foundations of Robotics VIII*, pages 467–481. Springer-Verlag, 2010.

- [42] S. Dalibard, A. Nakhaei, F. Lamiroux, and J.-P. Laumond. Whole-body task planning for a humanoid robot: a way to integrate collision avoidance. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 355–360, 2009.
- [43] T. Davies and A. Jnifene. Multiple waypoint path planning for a mobile robot using genetic algorithms. In *Proc. IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems (CIMS)*, pages 21–26, 2006.
- [44] D. Devalarazu and D. W. Watson. Path planning for altruistically negotiating processes. In *Proc. International Symposium on Collaborative Technologies and Systems (CTS)*, pages 196–202, 2005.
- [45] D. Devaurs, L. Bouard, M. Vaisset, C. Zanon, I. Al-Bluwi, R. Iehl, T. Siméon, and J. Cortés. MoMA-LigPath: A web server to simulate protein-ligand unbinding. *Nucleic Acids Research*, 41(W1):w297–w302, 2013.
- [46] D. Devaurs, L. Bouard, M. Vaisset, C. Zanon, I. Al-Bluwi, R. Iehl, T. Siméon, and J. Cortés. MoMA-LigPath: A web server to simulate protein-ligand unbinding. In *Proc. Journées Ouvertes en Biologie, Informatique et Mathématiques (JOBIM)*, 2013.
- [47] D. Devaurs, A. Shehu, T. Siméon, and J. Cortés. Sampling-based methods for a full characterization of energy landscapes of small peptides. In *Proc. IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2014.
- [48] D. Devaurs, T. Siméon, and J. Cortés. Parallelizing RRT on distributed-memory architectures. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2261–2266, 2011.
- [49] D. Devaurs, T. Siméon, and J. Cortés. Enhancing the Transition-based RRT to deal with complex cost spaces. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4105–4110, 2013.
- [50] D. Devaurs, T. Siméon, and J. Cortés. Parallelizing RRT on large-scale distributed-memory architectures. *IEEE Transactions on Robotics*, 29(2):571–579, 2013.
- [51] D. Devaurs, T. Siméon, and J. Cortés. Efficient sampling-based approaches to optimal path planning in complex cost spaces. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2014.
- [52] 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 International Conference on Intelligent Robots and Systems (IROS)*, 2014.
- [53] D. Devaurs, M. Vaisset, T. Siméon, and J. Cortés. A multi-tree approach to compute transition paths on energy landscapes. In *Proc. Workshop on Artificial Intelligence and Robotics Methods in Computational Biology (AAAI)*, 2013.
- [54] A. Dobson and K. E. Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *The International Journal of Robotics Research*, 33(1):18–47, 2014.
- [55] M. F. Dunn. Zinc-ligand interactions modulate assembly and stability of the insulin hexamer - a review. *Biometals*, 18(4):295–303, 2005.
- [56] J. M. Esposito, J. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In M. Erdmann, D. Hsu, M. Overmars, and A. F. van der Stappen, editors, *Algorithmic Foundations of Robotics VI*, pages 107–121. Springer-Verlag, 2005.

- [57] A. Ettlín and H. Bleuler. Randomised rough-terrain robot motion planning. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5798–5803, 2006.
- [58] A. Ettlín and H. Bleuler. Rough-terrain robot motion planning based on obstacle-ness. In *Proc. International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2006.
- [59] D. Ferguson, N. Kalra, and A. Stentz. Replanning with RRTs. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1243–1248, 2006.
- [60] D. Ferguson and A. Stentz. Anytime RRTs. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5369–5375, 2006.
- [61] D. Ferguson and A. Stentz. Anytime, dynamic planning in high-dimensional search spaces. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1310–1315, 2007.
- [62] J. Fink, N. Michael, S. Kim, and V. Kumar. Planning and control for cooperative manipulation and transportation with aerial robots. *The International Journal of Robotics Research*, 30(3):324–334, 2011.
- [63] D. Flavigné and M. Taïx. Improving motion planning in weakly connected configuration spaces. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5900–5905, 2010.
- [64] I. Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley, 1995.
- [65] D. Frenkel and B. Smit. *Understanding Molecular Simulations: From Algorithms to Applications*, volume 1 of *Computational Science Series*. Academic Press, 2nd edition, 2001.
- [66] M.-P. Gaigeot. Unravelling the conformational dynamics of the aqueous alanine dipeptide with first-principle molecular dynamics. *The Journal of Physical Chemistry B*, 113(30):10059–10062, 2009.
- [67] R. Gayle, K. R. Klinger, and P. G. Xavier. Lazy Reconfiguration Forest (LRF) - an approach for motion planning with multiple tasks in dynamic environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1323, 2007.
- [68] R. Geraerts and M. H. Overmars. Creating high-quality roadmaps for motion planning in virtual environments. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4355–4361, 2006.
- [69] R. Geraerts and M. H. Overmars. Creating high-quality paths for motion planning. *The International Journal of Robotics Research*, 26(8):845–863, 2007.
- [70] M. Gharbi. *Motion planning and object manipulation by humanoid torsos*. PhD thesis, LAAS-CNRS, Toulouse, 2010.
- [71] B. Gipson, D. Hsu, L. E. Kavraki, and J.-C. Latombe. Computational models of protein kinematics and dynamics: Beyond simulation. *Annual Review of Analytical Chemistry*, 5:273–291, 2012.

- [72] A. Grama, A. Gupta, G. Karypis, and V. Kumar. *Introduction to Parallel Computing*. Pearson Education, 2nd edition, 2003.
- [73] D. Guieysse, J. Cortés, S. Puech-Guenot, S. Barbe, V. Lafaquiére, P. Monsan, T. Siméon, I. André, and M. Remaud-Siméon. A structure-controlled investigation of lipase enantioselectivity by a path-planning approach. *ChemBioChem*, 9(8):1308–1317, 2008.
- [74] D. Henrich. Fast motion planning by parallel processing - a review. *Journal of Intelligent and Robotic Systems*, 20(1):45–69, 1997.
- [75] D. Hsu, R. Kindel, J.-C. Latombe, and S. M. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.
- [76] S. Huo and J. E. Straub. The MaxFlux algorithm for calculating variationally optimized reaction paths for conformational transitions in many body systems at finite temperature. *The Journal of Chemical Physics*, 107(13):5000–5006, 1997.
- [77] S. Huo and J. E. Straub. Direct computation of long time processes in peptides and proteins: reaction path study of the coil-to-helix transition in polyalanine. *Proteins: Structure, Function and Genetics*, 36(2):249–261, 1999.
- [78] J. Ichnowski and R. Alterovitz. Parallel sampling-based motion planning with super-linear speedup. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1206–1212, 2012.
- [79] R. Iehl, J. Cortés, and T. Siméon. Costmap planning in high dimensional configuration spaces. In *Proc. IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 166–172, 2012.
- [80] B. Isralewitz, M. Gao, and K. Schulten. Steered molecular dynamics and mechanical functions of proteins. *Current Opinion in Structural Biology*, 11(2):224–230, 2001.
- [81] S. A. Jacobs, N. Stradford, C. Rodriguez, S. Thomas, and N. M. Amato. A scalable distributed RRT for motion planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 5073–5080, 2013.
- [82] L. Jaillet, F. J. Corcho, J.-J. Pérez, and J. Cortés. Randomized tree construction algorithm to explore energy landscapes. *Journal of Computational Chemistry*, 32(16):3464–3474, 2011.
- [83] L. Jaillet, J. Cortés, and T. Siméon. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics*, 26(4):635–646, 2010.
- [84] L. Jaillet, A. Yershova, S. M. LaValle, and T. Siméon. Adaptive tuning of the sampling domain for dynamic-domain RRTs. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2851–2856, 2005.
- [85] J. Jájá. *An Introduction to Parallel Algorithms*. Addison-Wesley, 1992.
- [86] M. S. Jamal, S. Parveen, M. A. Beg, M. Suhail, A. G. Chaudhary, G. A. Damanhour, A. M. Abuzenadah, and M. Rehan. Anticancer compound plumbagin and its molecular targets: a structural insight into the inhibitory mechanisms using computational approaches. *PLoS ONE*, 9(2):e87309, 2014.

- [87] J. Jeon, S. Karaman, and E. Frazzoli. Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*. In *Proc. IEEE Conference on Decision and Control (CDC)*, pages 3276–3282, 2011.
- [88] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1470–1477, 2011.
- [89] E. Kahale, P. Castillo, and Y. Bestaoui. Minimum time reference trajectory generation for an autonomous quadrotor. In *Proc. International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 126–133, 2014.
- [90] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Proc. Robotics: Science and Systems (RSS)*, 2010.
- [91] S. Karaman and E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *Proc. IEEE Conference on Decision and Control (CDC)*, pages 7681–7687, 2010.
- [92] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for a class of pursuit-evasion games. In D. Hsu, V. Isler, J.-C. Latombe, and M. Lin, editors, *Algorithmic Foundations of Robotics IX*, pages 71–87. Springer, 2011.
- [93] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.
- [94] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the RRT*. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1478–1483, 2011.
- [95] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
- [96] S. Kiesel, E. Burns, C. Wilt, and W. Ruml. Integrating vehicle routing and motion planning. In *Proc. International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.
- [97] M. Kim, S.-H. Choi, J. Kim, K. Choi, J.-M. Shin, S.-K. Kang, Y.-J. Choi, and D. H. Jung. Density-based clustering of small peptide conformations sampled from a molecular dynamics simulation. *Journal of Chemical Information and Modeling*, 49(11):2528–2536, 2009.
- [98] Y. Kim, S. Choi, and W. Y. Kim. Efficient basin-hopping sampling of reaction intermediates through molecular fragmentation and graph theory. *Journal of Chemical Theory and Computation*, 10(6):2419–2426, 2014.
- [99] S. Kirillova, J. Cortés, A. Stefaniu, and T. Siméon. An NMA-guided path planning approach for computing large-amplitude conformational changes in proteins. *Proteins: Structure, Function, and Bioinformatics*, 70(1):131–143, 2008.
- [100] J. J. Kuffner and S. M. LaValle. RRT-Connect: an efficient approach to single-query path planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, 2000.

- [101] V. Lafaquière, S. Barbe, S. Puech-Guenot, D. Guieysse, J. Cortés, P. Monsan, T. Siméon, I. André, and M. Remaud-Siméon. Control of lipase enantioselectivity by engineering the substrate binding site and access channel. *ChemBioChem*, 10(17):2760–2771, 2009.
- [102] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [103] S. M. LaValle. Rapidly-exploring Random Trees: a new tool for path planning. Technical Report TR: 98-11, Computer Science Dept., Iowa State University, 1998.
- [104] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
- [105] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.
- [106] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A. K. Peters, Wellesley, MA, 2001.
- [107] J. Lee, C. Pippin, and T. R. Balch. Cost based planning with RRT in outdoor environments. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 684–689, 2008.
- [108] T.-Y. Li and Y.-C. Shie. An incremental learning approach to motion planning with roadmap management. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3411–3416, 2002.
- [109] S. R. Lindemann and S. M. LaValle. Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3251–3257, 2004.
- [110] T. Lozano-Pérez. Spatial planning: a configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.
- [111] S. K. Lüdemann, V. Lounnas, and R. C. Wade. How do substrates enter and products exit the buried active site of cytochrome P450cam? 1. Random expulsion molecular dynamics investigation of ligand access channels and mechanisms. *Journal of Molecular Biology*, 303(5):797–811, 2000.
- [112] R. Luna, I. A. Şucan, M. Moll, and L. E. Kavraki. Anytime solution optimization for sampling-based motion planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 5068–5074, 2013.
- [113] T. Maddula, A. A. Minai, and M. M. Polycarpou. Multi-target assignment and path planning for groups of UAVs. In S. Butenko, R. Murphey, and P. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization*, volume 3 of *Cooperative Systems*, pages 261–272. Springer, 2002.
- [114] J. Mainprice, E. A. Sisbot, L. Jaillet, J. Cortés, R. Alami, and T. Siméon. Planning human-aware motions using a sampling-based costmap planner. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 5012–5017, 2011.
- [115] M. Manubens, D. Devaurs, L. Ros, and J. Cortés. A motion planning approach to 6-D manipulation with aerial towed-cable systems. In *Proc. International Micro Air Vehicle Conference and Flight Competition (IMAV)*, 2013.

- [116] M. Manubens, D. Devaurs, L. Ros, and J. Cortés. Motion planning for 6-D manipulation with aerial towed-cable systems. In *Proc. Robotics: Science and Systems (RSS)*, 2013.
- [117] J. D. Marble and K. E. Bekris. Towards small asymptotically near-optimal roadmaps. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2557–2562, 2012.
- [118] J. D. Marble and K. E. Bekris. Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE Transactions on Robotics*, 29(2):432–444, 2013.
- [119] T. G. McGee and J. K. Hedrick. Path planning and control for multiple point surveillance by an unmanned aircraft in wind. In *Proc. American Control Conference (ACC)*, pages 4261–4266, 2006.
- [120] J. McMahan and E. Plaku. Sampling-based tree search with discrete abstractions for motion planning with dynamics and temporal logic. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.
- [121] N. Michael, J. Fink, and V. Kumar. Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 30(1):73–86, 2011.
- [122] M. Morales, S. Rodríguez, and N. M. Amato. Improving the connectivity of PRM roadmaps. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4427–4432, 2003.
- [123] R. M. Murray. Trajectory generation for a towed cable system using differential flatness. In *Proc. IFAC World Congress*, 1996.
- [124] D. Nieuwenhuisen and M. H. Overmars. Useful cycles in probabilistic roadmap graphs. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 446–452, 2004.
- [125] H. Okumura and Y. Okamoto. Temperature and pressure dependence of alanine dipeptide studied by multibaric-multithermal molecular dynamics simulations. *The Journal of Physical Chemistry B*, 112(38):12038–12049, 2008.
- [126] B. S. Olson and A. Shehu. Evolutionary-inspired probabilistic search for enhancing sampling of local minima in the protein energy surface. *Proteome Science*, 10(Suppl 1):S5, 2012.
- [127] M. Otte and N. Correll. Any-com multi-robot path-planning: maximizing collaboration for variable bandwidth. In *Proc. International Symposium on Distributed Autonomous Robotics Systems (DARS)*, 2010.
- [128] G. H. Paine and H. A. Scheraga. Prediction of the native conformation of a polypeptide by a statistical-mechanical procedure. III. Probable and average conformations of enkephalin. *Biopolymers*, 26(7):1125–1162, 1987.
- [129] J. Pan, C. Lauterbach, and D. Manocha. G-planner: real-time motion planning and global navigation using GPUs. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, pages 1245–1251, 2010.
- [130] C. Park, J. Pan, and D. Manocha. Real-time optimization-based planning in dynamic environments using GPUs. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4075–4082, 2013.

- [131] A. Perez, S. Karaman, A. Shkolnik, E. Frazzoli, S. Teller, and M. R. Walter. Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4307–4313, 2011.
- [132] R. Piechnick, E. Ritter, P. W. Hildebrand, O. P. Ernst, P. Scheerer, K. P. Hofmann, and M. Heck. Effect of channel mutations on the uptake and release of the retinal ligand in opsin. *Proceedings of the National Academy of Sciences of the United States of America*, 109(14):5247–5252, 2012.
- [133] E. Plaku. Planning in discrete and continuous spaces: From LTL tasks to robot motions. In G. Herrmann, M. Studley, M. Pearson, A. Conn, C. Melhuish, M. Witkowski, J. Kim, and P. Vadakkepat, editors, *Advances in Autonomous Robotics*, pages 331–342. Springer, 2012.
- [134] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-based roadmap of trees for parallel motion planning. *IEEE Transactions on Robotics*, 21(4):597–608, 2005.
- [135] E. Plaku and L. E. Kavraki. Distributed sampling-based roadmap of trees for large-scale motion planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3868–3873, 2005.
- [136] E. Plaku and L. E. Kavraki. Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In S. Akella, N. M. Amato, W. Huang, and B. Mishra, editors, *Algorithmic Foundations of Robotics VII*, pages 3–18. Springer-Verlag, 2008.
- [137] M. J. Quinn. *Parallel Computing: Theory and Practice*. McGraw-Hill, 1994.
- [138] S. Rathinam, R. Sengupta, and S. Darbha. A resource allocation algorithm for multivehicle systems with nonholonomic constraints. *IEEE Transactions on Automation Science and Engineering*, 4(1):98–104, 2007.
- [139] S. Rodríguez, X. Tang, J.-M. Lien, and N. M. Amato. An obstacle-based rapidly-exploring random tree. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 895–900, 2006.
- [140] D. S. Sauriyal, A. S. Jaggi, and N. Singh. Extending pharmacological spectrum of opioids beyond analgesia: Multifunctional aspects in different pathophysiological states. *Neuropeptides*, 45(3):175–188, 2011.
- [141] K. Savla, E. Frazzoli, and F. Bullo. On the point-to-point and traveling salesperson problems for Dubins vehicle. In *Proc. American Control Conference (ACC)*, pages 786–791, 2005.
- [142] S. Sengupta. A parallel randomized path planner for robot navigation. *International Journal of Advanced Robotic Systems*, 3(3):259–266, 2006.
- [143] A. Shehu. Probabilistic search and optimization for protein energy landscapes. In S. Aluru and M. Singh, editors, *Handbook of Computational Molecular Biology*, Computer & Information Science Series. Chapman & Hall/CRC, 2nd edition, 2013. in press.
- [144] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani. Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):729–746, 2004.

- [145] T. Siméon, J.-P. Laumond, and F. Lamiroux. Move3D: a generic platform for path planning. In *Proc. IEEE International Symposium on Assembly and Task Planning (ISATP)*, pages 25–30, 2001.
- [146] R. A. Skop and Y.-I. Choo. The configuration of a cable towed in a circular path. *Journal of Aircraft*, 8(11):856–862, 1971.
- [147] A. Steltzner, D. Kipp, A. Chen, D. Burkhart, C. Guernsey, G. Mendeck, R. Mitcheltree, R. Powell, T. Rivellini, M. San Martin, and D. Way. Mars Science Laboratory entry, descent, and landing system. In *Proc. IEEE Aerospace Conference*, 2006.
- [148] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3310–3317, 1994.
- [149] M. Strandberg. Augmenting RRT-planners with local trees. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3258–3262, 2004.
- [150] Z. Tang and Ü. Özgüner. Motion planning for multitarget surveillance with mobile sensor agents. *IEEE Transactions on Robotics*, 21(5):898–908, 2005.
- [151] C. Urmson and R. Simmons. Approaches for heuristically biasing RRT growth. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1178–1183, 2003.
- [152] J. van den Berg, D. Ferguson, and J. Kuffner. Anytime path planning and replanning in dynamic environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2366–2371, 2006.
- [153] H. Vashisth and C. F. Abrams. Ligand escape pathways and (un)binding free energy calculations for the hexameric insulin-phenol complex. *Biophysical Journal*, 95(9):4193–4204, 2008.
- [154] C. Velez-Vega, E. E. Borrero, and F. A. Escobedo. Kinetics and reaction coordinate for the isomerization of alanine dipeptide by a forward flux sampling protocol. *The Journal of Chemical Physics*, 130(22):225101:1–12, 2009.
- [155] D. J. Wales. *Energy Landscapes: Applications to Clusters, Biomolecules and Glasses*. Cambridge University Press, 2003.
- [156] D. J. Wales and J. P. Doye. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.
- [157] W. Wang, Y. Li, X. Xu, and S. X. Yang. An adaptive roadmap guided Multi-RRTs strategy for single query path planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2871–2876, 2010.
- [158] W. Wang, X. Xu, Y. Li, J. Song, and H. He. Triple RRTs: an effective method for path planning in narrow passages. *Advanced Robotics*, 24(7):943–962, 2010.
- [159] P. Williams. Optimal terrain-following for towed-aerial-cable sensors. *Multibody System Dynamics*, 16(4):351–374, 2006.
- [160] J. H. Yakey, S. M. LaValle, and L. E. Kavraki. Randomized path planning for link-ages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, 17(6):951–958, 2001.

-
- [161] G. Yang and V. Kapila. Optimal path planning for unmanned air vehicles with kinematic and tactical constraints. In *Proc. IEEE Conference on Decision and Control (CDC)*, pages 1301–1306, 2002.
- [162] A. Yershova and S. M. LaValle. Improving motion planning algorithms by efficient nearest-neighbor searching. *IEEE Transactions on Robotics*, 23(1):151–157, 2007.
- [163] R. Zhao, J. Shen, and R. D. Skeel. Maximum flux transition paths of conformational change. *Journal of Chemical Theory and Computation*, 6(8):2411–2423, 2010.
- [164] Q. Zhu, Y. Wu, G. Wu, and X. Wang. An improved anytime RRTs algorithm. In *Proc. International Conference on Artificial Intelligence and Computational Intelligence (AICI)*, pages 268–272, 2009.
- [165] R. Zhu, D. Sun, and Z. Zhou. Integrated design of trajectory planning and control for micro air vehicles. *Mechatronics*, 17(4-5):245–253, 2007.
- [166] M. Zucker, J. Kuffner, and M. Branicky. Multipartite RRTs for rapid replanning in dynamic environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1603–1609, 2007.