Proceedings of the
9th International Workshop on
Planning and Scheduling for Space (IWPSS)

Edited by

Steve Chien,
Marcelo Oglietti

IWPSS 2015 is held in conjunction with the
24th International Joint Conference on Artificial Intelligence.

Preface

This volume contains the papers presented at IWPSS 2015: 9th International Workshop on Planning and Scheduling for Space held on July 26-27, 2015 in Buenos Aires.


The proceedings will be archived at the robotics web site at ESTEC/ESA.

Steve Chien
Jet Propulsion Laboratory, California Institute of Technology

Marcelo Oglietti
Argentine Space Agency (CONAE)
# Table of Contents

Requirements-Based Scheduling for NASA’s Deep Space Network .......................... 1  
*Mark Johnston*

Augmented Motion Plans for Planning in Uncertain Terrains ............................. 2  
*Tsz-Chiu Au and Ty Nguyen*

Activity-based Scheduling of Science Campaigns for the Rosetta Orbiter .......... 8  
*Steve Chien, Gregg Rabideau, Daniel Tran, Martina Troesch, Joshua Doubleday, Federico Nespoli, Miguel Perez Ayucar, Marc Costa Sitja, Claire Vallat, Bernhard Geiger, Nico Altobelli, Manuel Fernandez, Fran Vallejo, Rafael Andres and Michael Kueppers*

Heuristic Onboard Pointing Re-scheduling for an Earth Observing Spacecraft .............................................................................................................. 9  
*Steve Chien and Martina Troesch*

BepiColombo Science Data Storage and Downlink Optimization Tool ............... 19  
*Sara de La Fuente, Nicola Policella, Simone Fratini and Jonathan McAuliffe*

Practical Goal Recognition for ISS Crew Activities ........................................... 29  
*Yolanda E-Martín, Maria D. R- Moreno and David Smith*

Resource Driven Timeline-Based Planning for Space Applications ................... 37  
*Simone Fratini, Nicola Policella, Nicolas Faerber, Andrea De Maio, Alessandro Donati and Bruno Sousa*

Mission Planning Systems for Commercial Small-Sat Earth Observation Constellations ........................................................................................................... 45  
*Claudio Iacopino, Simon Harrison and Andy Brewer*

A Constraint-based Optimizer for Scheduling Solar Array Operations on the International Space Station ................................................................. 53  
*Jan Jelínek and Roman Bartak*

A Constraint-based Planner for the Mars Express Orbiter ................................. 62  
*Martin Kolombo and Roman Bartak*

Postponing decision-making to deal with resource uncertainty on Earth-observation satellites ................................................................................................ 70  
*Adrien Maillard, Gérard Verfaillie, Cédric Pralet, Jean Jaubert, Isabelle Sebbag and Frédéric Fontanari*

Compiling Away Uncertainty in Strong Temporal Planning with Uncontrollable Durations ............................................................................................... 78  
*Andrea Micheli, Minh Do and David Smith*
Using Automated Scheduling for Mission Analysis and a Case Study
Using the Europa Clipper Mission Concept ......................................................... 87
  Gregg Rabideau, Steve Chien, Eric Ferguson
Heuristic Scheduling of Space Mission Downlinks: A Case study from
the Rosetta Mission .......................................................................................... 94
  Gregg Rabideau, Federico Nespoli and Steve Chien
SAOCOM Mission Planning Process: Combining Optimization and
Greedy Techniques ........................................................................................... 104
  Eduardo Romero
Automated Operator Link Assignment Scheduling for NASA’s Deep
Space Network .................................................................................................... 113
  Daniel Tran and Mark Johnston
An Algorithm for Generation of Antenna-Satellite Optimal Schedules
by using Mixed Integer Linear Programming ............................................... 120
  Rafael Vazquez, Ignacio Librero and Jorge Galan Vioque
Program Committee

Roman Bartak  Charles University in Prague
Steve Chien  Jet Propulsion Laboratory, California Institute of Technology
Mark Giuliano  Space Telescope Science Institute
Mark Johnston  Jet Propulsion Laboratory, California Institute of Technology
Russell Knight  Jet Propulsion Laboratory, California Institute of Technology
Bob Morris  NASA
Angelo Oddi  ISTC-CNR, Italian National Research Council
Marcelo Oglietti  CONAE
Nicola Policella  ESA/ESOC
Cédric Pralet  ONERA Toulouse
Robin Steel  VEGA
Requirements-Based Scheduling for
NASA’s Deep Space Network

Mark D. Johnston
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena CA USA 91109
mark.d.johnston @ jpl.nasa.gov

Abstract
This talk describes the Deep Space Network (DSN) scheduling engine component of a new scheduling system being deployed for NASA’s Deep Space Network. The DSE provides core automation functionality for scheduling the network, including the interpretation of scheduling requirements expressed by users, their elaboration into tracking passes, and the resolution of conflicts and constraint violations. The DSE incorporates both systematic search and repair-based algorithms, used for different phases and purposes in the overall system. It has been integrated with a web application that provides DSE functionality to all DSN users through a standard web browser, as part of a peer-to-peer schedule negotiation process for the entire network. The system has been deployed operationally and is in routine use, and is in the process of being extended to support long-range planning and forecasting as well as real-time DSN scheduling.

Overview
NASA’s Deep Space Network (DSN) provides communications and other services for planetary exploration missions as well as other missions beyond geostationary orbit, supporting both NASA and international users. It also constitutes a scientific facility in its own right, conducting radar investigations of the moon and planets, in addition to radio science and radio astronomy. The DSN comprises three antenna complexes in Goldstone, California; Madrid, Spain; and Canberra, Australia. Each complex contains one 70 meter antenna and several 34 meter antennas, providing S-, X-, and K-band up and downlink services. The distribution in longitude enables full sky coverage and generally provides some overlap in spacecraft visibility between the complexes.

Currently, the DSN scheduling process is centered around the Service Preparation Subsystem (SPS) which provides a central database for the real-time schedules and for the auxiliary data needed by the DSN to operate the antennas and communications equipment (e.g. viewperiods, sequence-of-events files). The ongoing project to improve scheduling automation is designated the Service Scheduling Software, or S³, and is an integrated component of SPS. There are three primary features of S³ that have significantly improved the DSN scheduling process:

1. Automated scheduling of activities with a request-driven approach (as contrasted with the previous activity-oriented approach that specified individual activities);
2. Unifying the scheduling software and databases into a single integrated suite covering realtime out through as much as several years into the future;
3. Development of a peer-to-peer collaboration environment for DSN users to view, edit, and negotiate schedule changes and conflict resolutions.

S³ is currently being deployed to schedule realtime DSN operations, as well as providing tools for forecasting and long-range planning.

Further details concerning S³, including the collaboration system and scheduling engine, may be found in the following references:

Augmented Motion Plans for Planning in Uncertain Terrains

Tsz-Chiu Au and Ty Nguyen
School of Electrical and Computer Engineering
Ulsan National Institute of Science and Technology
Ulsan, South Korea 689-798
{chiu, tynguyen}@unist.ac.kr

Abstract
In exploration of unknown planets, ground vehicles such as Mars rovers may not know exactly what terrain they will run into, causing great difficulty in meeting their goals. This paper presents a two-stage approach for motion planning in uncertain terrains. In the first stage, we utilize a specialized planner to generate motion plans to meet some arrival requirements. In the second stage, we augment the motion plans with sensing information and combine them to form a full-fledged controller in order to cope with uncertainty in the environment. This separation of planning and uncertainty management can simplify the development of planners for complicated goals. Our preliminary experiments showed that a vehicle can meet the arrival requirements with a high probability in small random graphs.

Introduction
Our next frontier of space exploration is near earth objects such as Mars and comets. Currently, we rely on robot rovers to explore these extraterrestrial lands. However, without full knowledge of the environment on these lands, it is hard to generate motion plans for these rovers to achieve their objectives. For example, if the vehicle in Figure 1 can only detect the paths within its sensors’ range (the orange pie shape), how can it arrive at the destination without knowing entire graph of feasible paths? In this paper, we focus on planning to move a vehicle or a rover to its destination in order to meet certain requirements at the destination, under the condition that the terrain on its way is uncertain.

There have been many works on motion planning under uncertainty in robotics (e.g., (Rekleitis, Meger, and Dudek 2006; Nakhaei and Lamiraux 2008; Um et al. 2013)), but few of them considered complex arrival requirements. Some variants of the popular probabilistic planning methods such as probabilistic roadmap (PRM) (Missiuro and Roy 2006; Kneebone and Dearden 2009) and rapidly-exploring random trees (Melchior and Simmons 2007; Maeda, Singh, and Durrant-Whyte 2011; Nikitenko et al. 2013) can be used to generate motion plans in uncertain environments. However, their underlying heuristics provide no guarantee of success. While the framework of POMDPs (e.g., (Ong et al. 2010)) is rich enough to encode planning problems under uncertainty with arbitrary utility functions, some goal conditions can be too complicated for general-purpose POMDP solvers to handle. In fact, without using specialized planners it is hard to satisfy certain goal conditions. We therefore propose a two-stage approach for motion planning under uncertainty: First, use an efficient planner to generate motion plans that meet the arrival requirements in uncertainty-free environments. Second, combine the solution plans generated by the planner to form a controller that can work in uncertain environments. Our approach is inspired by the work on the synthesis of strategies from interaction traces in (Au, Kraus, and Nau 2008), which outlined the foundation of our approach: by augmenting motion plans with simulated sensing...
information in some possible worlds, these motion plans can be combined to form a full-fledged controller that will meet the arrival requirements in these possible worlds. The separation of the planning and uncertainty management is a key advantage of our approach.

This paper is organized as follows. First, we define the motion planning problem in uncertain terrains. Second, we discuss how to combine the motion plans to cope with uncertain terrains. Our algorithm will then be evaluated experimentally with small random graphs, and the preliminary results will be presented. Finally, we go over some related work and summarize our contributions.

**Problem Definition**

We assume the terrain has a fixed set of roads, and off-road riding is not allowed. The road network can be modeled as a directed acyclic graph as shown in Figure 1, so that each road corresponds to an edge in the graph. The terrain is 3-dimensional, meaning that each road can have different slopes (see Figure 2). We assume that the homogeneity assumption hold in individual roads (i.e., the road condition would not change when the vehicle traverses the road). If a road does not satisfy the homogeneity assumption, we will model it as a sequence of edges in the graph such that each edge satisfies the homogeneity assumption.

As in (Au, Quinlan, and Stone 2012), we consider autonomous vehicles that are controlled by PID-controllers or other non-linear controllers whose control signals are setpoints, which are the target velocities for the vehicle. After setting a new setpoint \( \hat{v} \), the velocity will not change to \( \hat{v} \) immediately; instead it takes a while for the vehicle to settle down at \( \hat{v} \). For planning purposes, it is essential to know how long the vehicle takes to settle at \( \hat{v} \). We therefore rely on a performance model \( (T_{\text{stable}}, D_{\text{stable}}) \), which conservatively measures the time and distance the vehicle takes to settle at \( \hat{v} \) after changing the setpoint. More precisely, the stable time \( T_{\text{stable}}(v, \hat{v}) \) is the maximum time the vehicle takes to stabilize at \( \hat{v} \), and the stable distance \( D_{\text{stable}}(v, \hat{v}) \) is the average distance the vehicle travels for a period of time \( T_{\text{stable}}(v, \hat{v}) \) after changing the setpoint. Performance models are subject to the road condition, and therefore they can be different on different roads. Assume that we have already obtained the performance model of every road, using the estimation method in (Au, Quinlan, and Stone 2012).

In longitudinal control, a vehicle moves along a road according to a time-dependent velocity function \( \hat{v}(t) \) called setpoint schedule, where \( \hat{v}(t) \) is the setpoint for the vehicle at time \( t \). (Au, Quinlan, and Stone 2012) has discussed at length how to generate a setpoint schedule to control a vehicle to arrive at a specific position on an one-dimensional path at a given arrival time and velocity. But this notion of motion plans is not sufficient for the traversal of a graph since a vehicle also needs to decide which outgoing edges the vehicle should choose when it reaches a node. We therefore extend the notion of setpoint schedules to include control signals which decide which path the vehicle should take in a directed graph.

Consider a vehicle traversing a directed acyclic graph \( G = (N, E) \), following a path starting at node \( n_0 \) and ending at node \( n_{\text{end}} \) (\( n_{\text{end}} \) is called the destination). Suppose the vehicle moves along the path using a setpoint schedule \( \hat{v}(\cdot) \). In this paper, we assume the controller can act at discrete time points only. Hence, \( \hat{v}(\cdot) \) can be represented as a sequence of pairs \( \langle (t_0, v_0), (t_1, v_1), \ldots, (t_n, v_n) \rangle \), which means that the vehicle should choose the setpoint \( v_i \) at time \( t_i \), for \( 0 \leq i \leq n \). We augment the setpoint schedule with information about which outgoing edges the vehicle should choose at each node as follows: a motion plan \( \pi \) is \( \langle (t_0, v_0), (t_1, a_1), \ldots, (t_n, a_n) \rangle \), where \( a_i = \langle v_i, e_i \rangle \) is the control signals called an action at time \( t_i \). An action \( a_i \) is a vector with two components: \( \langle v_i, e_i \rangle \), where \( v_i \) is the setpoint at time \( t_i \), and \( e_i \) is one of the outgoing edges of the node at which the vehicle is located at time \( t_i \). If the vehicle is not located at a node at time \( t_i \) or the node it locates at time \( t_i \) has no outgoing edges, \( e_i = \text{nil} \).

Given a graph \( G = (N, E) \), a starting time \( t_0 \), and a starting velocity \( v_0 \) at \( n_0 \), our goal is to generate a motion plan \( \pi \) such that the vehicle will reach the destination \( n_{\text{end}} \) while satisfying a goal condition \( \mathcal{G} \). This motion plan is, of course, subject to the speed limit \( v_{\text{max}} \) as well as physical constraints of the vehicle as described in the performance model \( (T_{\text{stable}}, D_{\text{stable}}) \) of each edge \( e \in E \). Formally, we define our problem as follows. A validation problem \( \mathcal{P}_{\text{valid}} \) is a 4-tuple \( \langle (t_0, v_0), \Gamma, v_{\text{max}}, \mathcal{G} \rangle \), where

- \( (t_0, v_0) \) is the initial configuration;  
- \( \Gamma = (G, \{(D_e, T_{\text{stable}} e, D_{\text{stable}} e)\}_e \in E) \) is the specification of the graph \( G = (N, E) \), where \( D_e \) is the length of the edge \( e \in E \), \( T_{\text{stable}} e \) is the stable time function of \( e \), and \( D_{\text{stable}} e \) is the stable distance function of \( e \);  
- \( v_{\text{max}} \) is the speed limit of all edges; and  
- \( \mathcal{G} \) is the goal condition.

We use the velocity function \( v(\cdot) \) to denote the velocity of the vehicle over time. We say \( v(\cdot) \) is constructible if there exists a setpoint schedule \( \hat{v}(\cdot) \) such that the velocity function is \( v(\cdot) \) if the vehicle follows \( \hat{v}(\cdot) \). Let \( \rho = (e_1, e_2, \ldots, e_m) \) be a path in \( G \) connecting \( n_0 \) to \( n_{\text{end}} \). A velocity function \( v(\cdot) \) is feasible for a path \( \rho \) if it satisfies the following constraints:

- \( C1 \) \( v(t_0) = v(0) = v_0 \);  
- \( C2 \) \( 0 \leq v(t) \leq v_{\text{max}} \) for \( 0 \leq t \leq t_{\text{end}} \), where \( t_{\text{end}} \) is the arrival time (i.e., the velocity cannot exceed the speed limit or be negative at any time);  
- \( C3 \) \( \int_{t_i}^{t_{i+1}} v(t) dt = D_{e_i} \) for all edge \( e_i \) on \( \rho \), where \( t_{e_i} \) is the time the vehicle reaches \( e_i \) according to \( v(\cdot) \) along \( \rho \) and \( t_{e_i+1} = t_{\text{end}} \) (i.e., the distance traveled on an edge \( e_i \) must be equal to the length \( D_{e_i} \) of \( e_i \));  
- \( C4 \) \( v(\cdot) \) is constructible; and  
- \( C5 \) \( \mathcal{G} \) is true.

A setpoint schedule \( \hat{v}(\cdot) \) is feasible for a path \( \rho \) if the velocity function constructed by \( \hat{v}(\cdot) \) is feasible for \( \rho \). A motion plan (say \( \pi = \langle (t_i, (v_i, e_i))\rangle_{i=0,n} \)) is feasible if the corresponding setpoint schedule \( \langle (t_i, v_i)\rangle_{i=0,n} \) is feasible for the corresponding path \( \langle (e_i, 0 \leq i \leq n \land e_i \neq \text{nil}) \rangle \). The objective of \( \mathcal{P}_{\text{valid}} \) is to decide whether a feasible motion plan exists. Notice that \( \mathcal{P}_{\text{valid}} \) has not yet taken uncertainty into account.
Motion Planning in Uncertain Terrains

This section concerns with situations in which the terrain is not fully observable. At the beginning, the vehicle knows nothing about the terrain beyond the range of its sensors, except the location of the destination nend. Suppose the vehicle starts with a belief about the set G of possible graphs. The belief is defined in terms of a probability distribution P over G, which means that the probability that a possible graph G ∈ G is the real graph is P(G). As the vehicle traverses the graph, it gathers more and more information about the terrain over time. This information will be helpful for the vehicle to ascertain which graph in G is real.

Let G = {G1, G2, ..., GmG} where Gi = (Ni, Ei) for 1 ≤ i ≤ mG. Some of the nodes and edges are shared by multiple graphs in G. Let N′ = ⋃1≤i≤mG {Ni} and E′ = ⋃1≤i≤mG {Ei}. The union of all graphs in G forms a supergraph G∗ = (N′, E′). Consider the 2D rectangular region R that physically contains G∗. We subdivide the region into a l × l grid as shown in Figure 1. Each cell in the grid will generate a signal when the sensors on the vehicle gather information about the cell. The signal of a cell reflects some features of the terrain at the cell (e.g., landmarks). From the sensors’ viewpoint, a terrain is a mapping T : [1...l] × [1...l] → S, which specifies the signals at all cells in R, where S is the set of all possible signals. Each possible graph Gi ∈ G is associated with a terrain Ti. Suppose the real graph is Gi. At the beginning, the vehicle has only partial knowledge of Ti. The vehicle will use its sensors to gather more information about Ti during traversal. We will make use of the monotonicity assumption: Any new knowledge from sensors will not contradict the existing ones.

Suppose the vehicle follows a feasible motion plan π = ⟨(t0, a0), (t1, a1), ..., (tn, an)⟩, where ai = ⟨vi, ei⟩. We assume sensing actions interleave with the execution of actions: Before the execution of an action ai at time ti, the vehicle obtains a percept bi from the environment. We define an augmented motion plan as τ = ⟨(a1, b1), (a2, b2), ..., (an, bn)⟩, which is basically an interaction trace between the vehicle and the environment. Now we make use of the results in (Au, Kraus, and Nau 2008), which states that a set of interaction traces, under certain conditions, can be combined to form an agent that will succeed in environments in which the interaction traces are generated. Furthermore, if we carefully select the interaction traces, we can increase the probability that the agent will succeed in an uncertain environment.

Algorithm 1 The architecture of the vehicle’s controller.

1: procedure VEHICLECONTROLLER(τ)
2:   i := 1; τi := τ
3: while the vehicle has not reached nend do
4:   if the current time is ti then
5:     Obtain a percept bi from sensors.
6:     τi+1 := ∅
7:     for all τ ∈ τi do
8:       if bi = percepti(τ) then
9:         τi+1 := τi+1 ∪ {τ}
10:     Ai := {actioni(τ) : ∀τ ∈ τi+1}
11:     if |Ai| ≠ 1 then
12:       Return Fail since τ is not compatible.
13:     Let the unique action in Ai be (vi, ei)
14:     if ei ≠ nil then steer the vehicle to edge ei.
15:     Change the current setpoint to vi
16:     i := i + 1

k = kb < ka, or (3) a′k+1 ≠ a′k.

Definition 2 Two augmented motion plans τ1 and τ2 are compatible if and only if \( |\text{lcp}(\text{action}(τ_1), \text{action}(τ_2))| > |\text{lcp}(\text{percept}(τ_1), \text{percept}(τ_2))| \), where action(⟨(a_i, b_i)⟩) = ⟨a_i⟩_{i=1..k} and percept(⟨(a_i, b_i)⟩) = ⟨b_i⟩_{i=1..k}.

Definition 3 A set τ of augmented motion plans is compatible if and only if every pair of augmented motion plans in τ is compatible.

Theorem 1 states that if the input τ of Algorithm 1 is compatible, the algorithm will never return Fail from Line 12. The proof of the theorem is similar to the proof in Theorem 2 in (Au, Kraus, and Nau 2008).

Theorem 1 Algorithm 1 will not fail if 1) τ is compatible, and 2) one of the augmented motion plans in τ is generated by the real graph.

Theorem 1 implies that if we can find a compatible set τ of feasible augmented motion plans, Algorithm 1 will always be able to reach its destination while satisfying the goal condition, as long as the real graph is one of the graphs in which some feasible augmented motion plans in τ are generated. Of course, we assume that the underlying planner for the validation problem can generate motion plans that satisfy the goal condition. For instance, if we concern with arriving at the destination at a specific time and at a specific velocity, the algorithms presented in (Au, Kraus, and Nau 2008) will work.

The probability of success of Algorithm 1 is equal to \( \sum_{τ \in τ} P(G_i) \), where τi is a feasible augmented motion plan generated in Gi. As can be seen, if τ includes one augmented motion plan in every G ∈ G, the probability of success is 1—the vehicle can guarantee to arrive at the destination in uncertain terrains. However, not every pair of augmented motion plans is compatible. In fact, there can be two possible graphs whose sets of all feasible augmented motion plans are disjoint, meaning that it is impossible to have a 100% successful rate. In these uncertain terrains, we can only hope to maximize the probability of success by finding
Hence, we present a greedy algorithm to find such \( T \). The algorithm, as shown in Algorithm 2, considers the possible graph in the descending order of probability, and then randomly generates \( K \) augmented motion plans in these graphs. The augmented motion plan will be added to \( T \) as long as it is compatible with all plans in \( T \). Although the algorithm cannot guarantee to find \( T \) that maximizes the probability of success, it can often find a good set of augmented motion plans with a high probability of success, as shown in the experimental results in the next section.

**Preliminary Experimental Results**

To evaluate Algorithm 1 and Algorithm 2, we conducted a simulation experiment with four different random graphs. One of the random graphs is shown in Figure 3. First, a directed acyclic graph \( G' \) that connects \( n_0 \) to \( n_{\text{end}} \) was generated by randomly connecting a fixed number of nodes (Graph a). \( G' \) has to be solvable (i.e., there exist paths connecting \( n_0 \) to \( n_{\text{end}} \)). Second, we randomly chose the road condition \( T_{\text{stable}} \) and \( D_{\text{stable}} \) for each edge, and assigned landmarks to each of the edges, such that the landmarks on two different edges are different. Third, we randomly removed some edges in \( G' \) to form graphs (Graph b-e). The landmarks on the removed edges were also removed. We repeated the removal of edges from \( G' \) four times to generate a set \( G \) of four possible graphs.

After generating \( G \), we set the starting time \( t_0 = 0s \) and the starting velocity \( v_0 = 0m/s \). Our goal \( G \) is to reach \( n_{\text{end}} \) at time \( t_{\text{end}} = 100s \) and \( v_{\text{end}} = 40m/s \). We devised a motion planning algorithm to generate one augmented motion plan for each possible graph. Then we randomly assigned a probability distribution \( P \) over \( G \), and ran Algorithm 2 to find a set \( T' \) of compatible augmented motion plans. We ran Algorithm 1 with \( T' \) 100 times and measured the probability of success. We repeated the measurement 6 times with different probability distributions. The successful rates and their 95\% confidence intervals are shown in Table 1.

As can be seen, the success rate of our approach is often higher than 90\%, with an overall successful rate of 97.4\%. While different probability distributions of \( G \) produce similar successful rates, the successful rates heavily depend on the topology of the original supergraph \( G' \). In \( G_5 \), the successful rates are 100\%, meaning that the augmented motion plans of the possible graphs are highly compatible.

**Related Work**

Probabilistic roadmap methods (PRM) (Kavaki et al. 1996) and rapidly-exploring random trees (LaValle and James J. Kuffner 2000) are both widely used, sampling-based algorithms. These algorithms are incomplete, but some extensions have been made to turn them into complete algorithms. Hirsch and Halperin (2004) and Zhang et al. (2007) proposed a hybrid motion planner that generates complete solutions with PRM. Nonetheless, these modified algorithms will suffer from inefficiency due to their completeness, and the arrival requirement has to be quite simple (e.g., arrive at a position without time and velocity requirement). While interleaving planning and execution is a good strategy to deal with uncertainty in planning (e.g., (Pivtoraiiko, Mollinger, and Kumar 2013)), replanning cannot correct wrong decisions in previous steps, thus it is hard to provide any arrival guarantees.

TPOPEXEC (Muise, Beck, and McIlraith 2013) introduces a two-stage approach for planning: First, an offline preprocessor takes a partial-order plan and a set of temporal
constraints to produce a generalized representation. Second, an online component called EXECUTOR selects a temporally consistent, valid plan fragment from the generalized plan. Choset et al. (2000) presented a sensor-based motion planning approach based on a roadmap called hierarchical generalized Voronoi graph (HGVG), which can be incrementally constructed using only line-of-sight sensor data. This approach guarantees that the robot can find a path from start to goal or report that such a path is not feasible. Luna et al. (2014) introduces a two-stage framework for efficient computation of an optimal control policy in the presence of uncertainty. It first generates a bounded-parameter Markov decision process (BMDP) over a discretization of the environment and then quickly selects a local policy within each region to optimize a continuously valued reward function online. As the sensors gather more information about the environment, the BMDP is updated accordingly and the global control policy is recomputed. However, none of the above approaches concern with arrival requirements other than reaching the destination.

**Concluding Remarks**

In this paper, we proposed a two-stage approach for motion planning in uncertain terrains. More specifically, we proposed to augment motion plans with sensing information and then combine them, in a greedy manner, to form a controller that can handle uncertainty in the environment. This separation of planning and uncertainty management can greatly simplify our task, as we can utilize existing fast planners to generate motion plans that satisfy the goal conditions. In space exploration applications, the goal conditions can be quite specific. Instead of modifying existing planning algorithms for these applications to deal with uncertain terrains, we propose to adopt them directly and combine their solutions to form a contingency controller. If there are enough augmented motion plans, the controller can achieve a high success rate.

Our approach has two drawbacks. First, the number of contingencies can be quite large in an uncertain environment, meaning that we may need a lot of augmented motion plans in order to deal with all these contingencies. However, we believe that in some environments, a small number of augmented motion plans is sufficient because one augmented motion plan can deal with several different contingencies in different possible graphs. We intend to evaluate this possibility in the future. Second, we usually do not know the set of all possible graphs in $\mathcal{G}$ ahead of time, thus some possible graphs are not considered by Algorithm 2. As discussed in (Au, Kraus, and Nau 2008), we can use a backup planner to handle these unknown cases. In the future, we intend to improve our algorithms to address these issues.

**References**


