MISSION-DIRECTED PATH PLANNING FOR PLANETARY ROVER EXPLORATION

Paul Tompkins

CMU-RI-TR-05-20

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

May 13, 2005

*Submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy*

Abstract

Robotic rovers uniquely benefit planetary exploration - they enable regional exploration with the precision of in-situ measurements, a combination impossible from an orbiting spacecraft or fixed lander. Current rover mission planning activities utilize sophisticated software for activity planning and scheduling, but simplified path planning and execution approaches tailored for localized operations to individual targets. Routes are coarsely hand-selected by human operators and executed by the rover's local obstacle detection and avoidance software. Neither route selection nor navigation deeply considers high level mission goals, large scale terrain, time, resources or operational constraints.

This strategy is insufficient for the investigation of multiple, regionally distributed targets in a single command cycle. Path planning tailored for this task must consider the impact of large scale terrain on power, speed and regional access; the effect of route timing on resource availability; the limitations of finite resource capacity and other operational constraints on vehicle range and timing; and the mutual influence between traverses and upstream and downstream stationary activities. Encapsulating this reasoning in an efficient autonomous planner would allow a rover to continue operating rationally despite significant deviations from an initial plan.

This research presents mission-directed path planning that enables an autonomous, strategic reasoning capability for robotic explorers. Planning operates in a space of position, time and energy. Unlike previous hierarchical approaches, it treats these dimensions simultaneously to enable globally-optimal solutions. The approach calls on a new incremental search algorithm designed for planning and re-planning under global constraints, in spaces of higher than two dimensions. Solutions under this method specify routes that avoid terrain obstacles, optimize the collection and use of rechargable energy, satisfy local and global mission constraints, and account for the time and energy of interleaved mission activities. Furthermore, the approach efficiently re-plans in response to updates in vehicle state and world models, and is well suited to online operation aboard a robot.

Simulations exhibit that the new methodology succeeds where conventional path planners would fail. Three planetary-relevant field experiments demonstrate the power of mission-directed path planning in directing actual exploration robots. Offline mission-directed planning sustained a solar-powered rover in a 24-hour sun-synchronous traverse. Online planning and re-planning enabled full navigational autonomy of over 1 kilometer, and supported the execution of science activities distributed over hundreds of meters.

Acknowledgements

In the pursuit of this research, I have been fortunate to associate with so many intelligent, motivating and helpful individuals. I begin by thanking my advisor, Red Whittaker. He inspired me early to pursue the grandest of dreams, and pushed me to work harder than I ever had. I reflect proudly on our time interacting with scientists, engineers and entrepreneurs to elevate and enable concepts for robotic lunar polar exploration. I maintain hope that one day we will succeed together in operating a robot on the Moon.

I would also like to thank Tony Stentz. His technical mentorship inspired the work I present in this document. Our partnership on the Advanced Global Path Planning project allowed me to mix with the space robotics community like I never had before. For his support of TEMPEST as leader of three successful robot developments and field campaigns, my warmest thanks also goes to David Wettergreen. Dave's competent yet easygoing leadership style and incredible diversity of knowledge in robotics and field work inspires me for my own career. Tony's and Dave's incredible energy fuels my belief that it *is* possible to succeed with career, family and friends simultaneously.

I could not have tested my work under relevant conditions without my fellow teammates on the Sun-Synchronous Navigation project and the Life in the Atacama project, including Vijay Baskaran, Bernardine Dias, Stu Heys, Dom Jonak, Ben Shamah, Trey Smith, Jim Teza, Chris Urmson, Vandi Verma, Dan Villa, Mike Wagner and Chris Williams. Their enormous effort and skill led to the creation of Hyperion and Zoe, two robots uniquely qualified to test long-distance, solar powered exploration strategies. Their friendships moulded field experiments into some of the most fun experiences of my life.

I thank my colleagues on the Advanced Mars Global Path Planning project, including Bernardine Dias for helping to integrate TEMPEST into the NASA CLARAty software repository, Ayorkor Mills-Tettey for her dedicated assistance in testing ISE, and Marc Zinck for creating an outstanding planning visualization tool.

I also want to thank my mountaineering friends in the Explorers Club of Pittsburgh, including Aaron Bennett, Tom Brooks, Bill Brose, Bob Coblentz, Jason DiChicchis, Shawn Klimek, Dave Micklo, Bill Molczan, and Mariann Mondik. Together we shared unforgettable experiences in some very cold, high places, and built friendships that will last forever. Their pull to get me away from research and into the wild places on this planet kept me sane through all my years at Carnegie Mellon.

This thesis would not have been possible without the love of my companion, Vandi Verma. We helped each other through the hard times over the months of thesis writing, and spent all our free moments plotting a way to live and work in the same city together. Now our theses are done, and every bit of planning for the future has paid off.

This thesis is dedicated to my parents, Mimi and David Tompkins, without whose unending confidence, support and love for me I could not have achieved this dream. I love you both.

Table of Contents

List of Figures

List of Tables

1. Introduction

Robotic rovers have been demonstrated as effective tools for planetary surface exploration on the moon [23] and on Mars [43]. As a result of early success with the Pathfinder and Mars Exploration Rover missions, NASA has projected follow-on Mars rover missions with increasing technological and scientific ambition. In the course of their development, these programs will lay the foundation for robotic technology that will enable access to a far greater range of locations on Mars and other bodies in the Solar System. One of the most exciting research thrusts is the development of robot navigational autonomy. Path planning and execution components allow a robot to select and navigate paths across planetary landscapes without human assistance. This thesis contends that to serve future missions, the scope of automated reasoning for navigation must include mission relevant parameters like time, resources, constraints and mission objectives. This research achieves significant advances in autonomous navigation that is cognizant of mission parameters and enables far more difficult surface operations than were previously possible.

1.1 Planetary Rover Navigational Autonomy

What will be demanded of rover navigational autonomy in future missions? Before creating a vision for future navigational autonomy, it is useful to assess the approaches taken in the most recent rover missions - the Mars Pathfinder mission and the combined Mars Exploration Rover missions - as well as a state-of-the-art research system. Over these three examples, note the clear disparity between the growing sophistication of automated stationary activity planning, and navigation planning, which continues to be restricted to obstacle avoidance.

1.1.1 Mars Pathfinder: Sojourner Rover

Sojourner made the first steps toward rover navigational autonomy on another planet [43]. Sojourner relied heavily on both the Pathfinder lander and a team of Earth-based engineers and scientists to enable travel to places of interest. The Pathfinder lander produced stereo imagery used to generate three-dimensional models of the landing site terrain. Human operators used a graphical user interface that combined the terrain model and a kinematic model of Sojourner to estimate safe routes of travel - routes that minimized the traversal of rock obstacles and avoided regions that prevented direct line-of-sight between Sojourner and the Pathfinder lander (and hence prevent communications and pose estimation via stereo vision). Operators selected waypoints along these safe paths, at intervals of 1-2 meters, as intermediate goals for autonomous navigation.

Using cameras and laser stripers, Sojourner executed “Go To Waypoint” commands by periodically assessing the difficulty of terrain ahead of the rover, and performing scripted avoidance maneuvers to circumnavigate obstacles. The rover avoided pursuing unreachable goals by abiding by a timeout clock that prevented travel after a set number of hours. Sojourner managed its resources during execution - it measured solar array current as a means of determining whether sufficient power was available for various activities. It also periodically checked its communications link with the lander, and executed a path reversal contingency action if communications were lost. Sols, or Martian days, were typically devoted to one type of activity - either traverse, or one of many possible science or engineering activities. Using this general approach, Sojourner covered more than 100 meters, all within 12 meters of the Pathfinder lander (see Figure 1-1), over 83 sols.

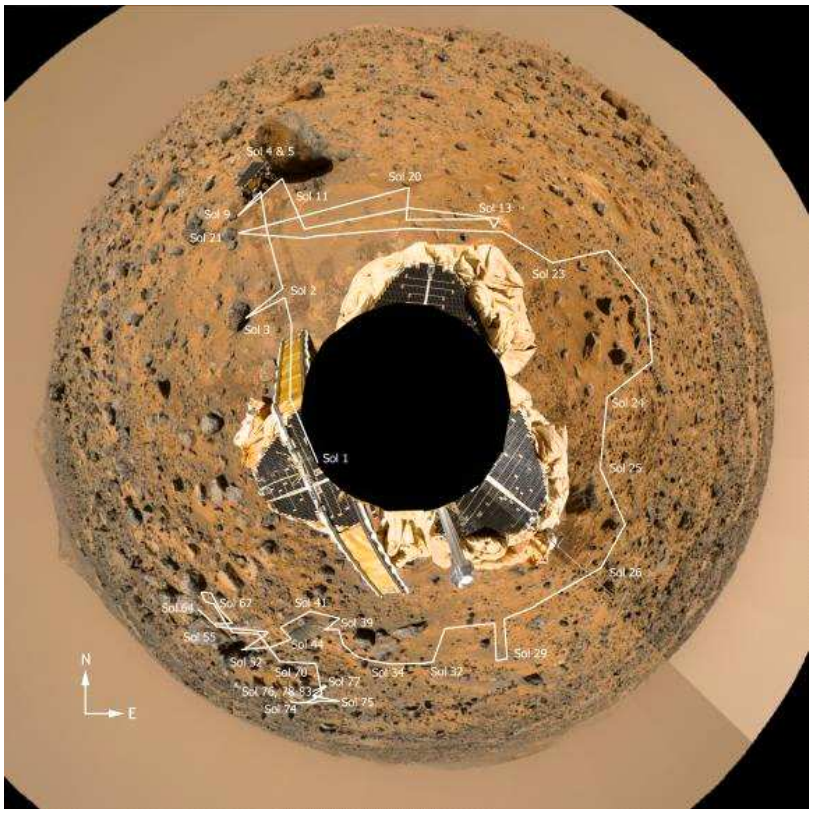


Figure 1-1: The Entire Pathfinder/Sojourner Mission Path

Summary of Pathfinder Observations:

- Human operators relied on a global model derived from Pathfinder lander stereo imagery for waypoint selection.
- Terrain traversability and communications line-of-sight geometric constraints were critical in selecting waypoints.
- Traverse activities were largely isolated from other focussed rover activities (e.g. science measurements), allowing waypoint selection and activity sequencing to occur mostly independently.
- Terrain, time, resources and communications remained a consideration in deciding the next course of action during traverse execution.
- Simple terrain sensing and scripted obstacle avoidance behaviors, combined with human operators' strong *a priori* knowledge of the terrain, enabled Sojourner to navigate confidently immediately around the lander. Sojourner's reliance on the lander for obstacle avoidance, state estimation and communications prevented it from travelling well beyond the landing site.

1.1.2 Mars Exploration Rovers: Spirit and Opportunity

The Mars Exploration Rover (MER) missions have far surpassed Pathfinder in autonomous operations on a planet. Spirit and Opportunity are independent of their landing vehicles, allowing them to traverse far from their landing sites. The MER rovers produce their own stereo imagery, both from hazard cameras (mounted at fixed angles on the rover) and the Pancam instrument (mounted on a mast pan/tilt mechanism). As with Sojourner, MER operators use a graphical user interface to assess the terrain around the rover, and hand-select waypoints that avoid hazardous terrain on the way to long-distance goals. Distant goals are selected using imagery collected from orbit. During the Martian winter months, when the sun was lowest on the horizon, rover operators were also forced to find paths and loiter points that maximized the solar array's exposure to sunlight. Travel favored sun-facing slopes, and slopes facing away from the sun were often removed from consideration.

Human operators must designate the navigation mode of the traverse - either "blind" whereby the rover drives in a straight path between waypoints without visual sensing, or in "autonomous navigation" mode that enables autonomous closed-loop driving. In a conservative strategy, blind mode driving is favored for the portion of a traverse nearest the rover where *a priori* stereo data is most reliable, and autonomous navigation mode is used to safeguard the rover from hazards where *a priori* data is least reliable.

Earth-based MER planning incorporates substantial autonomy. The MAPGEN system [2] combines a plan editing system called APGEN and automated reasoning derived from the EUROPA constraint-based planner [28]. Though plans remain largely hand-edited, EUROPA enables active constraint enforcement during the edit process, completes partial plans and repairs plans that violate constraints or resources, and provides operators with explanations for why certain edits are illegal.

During a traverse in autonomous navigation mode, the MER rovers create three-dimensional maps of the terrain at periodic intervals, and automatically segment the maps into traversability "goodness" levels. The GESTALT algorithm [19] evaluates the drive arcs available in the next move, and picks the best arc in terms of the goodness traversability index. At the time of publication of this document, Opportunity achieved a maximum autonomous drive segment of 85 meters on Sol 82 (the 82nd Martian day of operations), and Spirit, a segment of 78 meters on its Sol 133 [26].

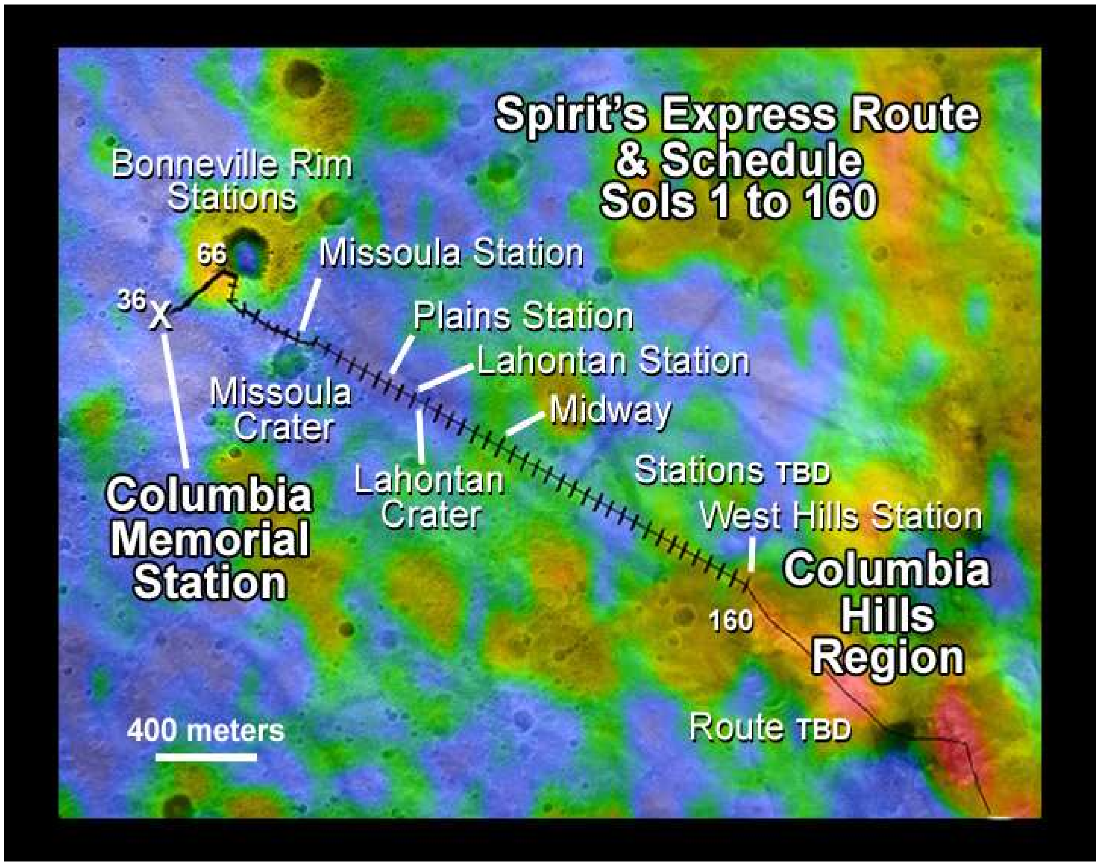


Figure 1-2: Spirit Traverse Route, Sol 1 to Sol 160. MER demonstrates the scientific interest in regional exploration. These distances might be traversed in a fraction of the time spent by Spirit, using greater levels of navigational autonomy.

Summary of MER Observations:

- Human operators use orbital imagery and local models derived from rover stereo imagery to manually select waypoints.
- Terrain traversability and sun, solar array and terrain geometry data are used to manually estimate the best path.

- Loosely interleaved traverse and science activities force a greater consideration of activity interaction than for Pathfinder. Activity planning is conducted on Earth, but with elements of autonomy. Traverse segments are inserted into activity plans using distance measurements between waypoints to estimate the duration and energy allocations for mobility segments.
- Spirit and Opportunity achieve waypoint goals autonomously, using local terrain evaluation. However, beyond consideration of the next drive arc, the rovers do not conduct planning to optimize travel to the next waypoint.

The MER missions illustrate the current state-of-art in flight rover operations. As of this writing, Spirit and Opportunity have operated for over three times their baseline lifetime of 90 sols. In 270 sols, they had covered a total of 3641 m and 1664 m, respectively and continue to operate productively. Yet mission planning remains a very labor-intensive activity, requiring extended support from large teams of experts.

1.1.3 Experimental State of the Art: CLEaR

High levels of autonomy have been achieved on orbiting spacecraft, as demonstrated by the Deep Space 1 Remote Agent Experiment [3]. However, the surface operations environment is far less predictable and, as described above, has historically required far more human oversight.

Recent Earth-based experiments have demonstrated limited autonomous surface science operations. As an example, Estlin *et al.* have developed a system for planning and execution of position-distributed science measurements [12]. The system comprises CASPER, a constraint-based iterative-repair planner [9], the TDL task-based executive [62], and a simple global path planner. The system processes are coordinated under the CLEaR framework, which delegates responsibilities between the planner and executive for plan repair in response to new data. Rover experiments on the Rocky 7 and 8 rovers in the JPL Mars Yard demonstrated the system's ability to devise plans given a set of high-level measurement goals distributed over several locations in rocky terrain. The planner decomposed the high-level goals into low-level activities, and selected the feasible subset of goals that maximized expected science return and respected resource constraints. In planning motion between measurement sites, CASPER called on the global path planner to estimate the distance of travel. Using models of the rover, CASPER then estimated the duration and energy requirements for the traverse, and integrated the traverse segment as a token in the activity plan.

During plan execution, the CLEaR system repaired plans in response to unanticipated data. The Morphin local navigation system [59] detected new obstacles that invalidated the original plans by requiring longer traverses between measurements. Measurement activities occasionally took longer to complete than anticipated. In such cases, the CLEaR system coordinated plan repair and removed lower-priority goals that could no longer be accomplished within the allotted time.

Summary of CLEaR Observations:

- Goal selection was performed by human operators and paths between goals were not pre-planned. However, activity planning considered the expected time and energy costs for traverse activities, which was derived from path planning that avoided all known obstacles.

- Traverse and science activities were closely interleaved, preventing an approach that planned them independently. The time and resource allocations of each activity strongly affected the feasibility of the others. Despite this, path planning to estimate the distance between goals was ignorant of mission goals, time, resource limitations and geometric constraints.

- Rocky 7, Rocky 8 and FIDO achieved the goal positions autonomously like the MER rovers. Rover navigation considered only the local terrain and immediate drive arcs. The eventual paths followed by the rovers had little or no connection to the paths generated to estimate the distance between goals in the planning phase.

The CLEaR example represents a major advance towards rover autonomy for localized operations. The test scenarios enabled automated planning and execution of a number of goals that would have occupied several sols of operations in Pathfinder or MER. For focussed site surveys, a system like CLEaR might enable a far greater collection of science data, and could significantly reduce the number of operational staff needed to oversee daily activities. However, for science operations distributed over greater distances, path planning would have to take a far more prominent role in planning and execution.

In the context of missions conducted at greater scales, with more complex terrain, dynamic lighting and resources that vary spatially and temporally, a plan that purely avoids terrain obstacles might, at best, be inefficient or operationally infeasible and, at worst, might endanger the rover and the mission.

1.2 Future Rover Scenarios

Future rover missions will demand far more coordination between activity planning and navigation planning. A greater ambition for distance, and pressure to reduce operations staff, will require navigational autonomy over greater distances - a schedule involving daily long-distance traverses cannot afford the labor of the detailed scrutiny seen in MER. Long distance traverses will intersect a variety of terrains, whose slope and orientation affect locomotion and solar power. Motion with respect to large-scale terrain features may entail driving through sunlight or communications shadows. Missions will take greater advantage of available time, operating from dawn to dusk, and even at night. Diurnal variations will affect power, thermal and sensing systems. Resources will continue to be in short supply. The navigation route, timing and resource profile will inevitably affect the preconditions for downstream activities, and vice versa. Under these projected circumstances, this thesis asserts the need for a new kind of path planning that considers these factors - mission-directed path planning.

The next subsections introduce two mission scenarios that directly motivate mission-directed path planning.

1.2.1 Mars Exploration

Mission-directed path planning is initially motivated by the needs of future missions to Mars, like Mobile Science Laboratory [11], Mars Sample Return, and others. These missions may send highly-capable rovers to Mars for multi-year scientific surveys covering tens of kilometers of terrain. Missions such as these would benefit from reliable, effective rover autonomy, which would ease the planning workload of human operators in support of the missions for years at a time, and which could make intelligent decisions about the use of time and resources in unexpected situations.

Figure 1-3: Future Mars missions will require rovers to access increasingly difficult terrain.

Judging from Mars Pathfinder and the Mars Exploration Rover missions, energy management will continue to be a prime concern for future missions. Solar power remains a technologically simple means of generating power on Mars. A solar powered rover must consider the energy cost of its path of motion and determine how the time of travel affects the orientation of its solar array relative to the sun in the sky. Terrain may cast shadows on the route, particularly at dawn and dusk. At the end of each sol, a rover must recharge its batteries in preparation for survival or limited night operations. A careful evaluation of hibernation sites may allow the rover to find slopes that receive sunlight earlier the following morning, and tip the solar arrays to improve solar power throughout a sol.

Nuclear thermoelectric power generation is gaining favor for rover power. The clear advantage of nuclear power is that it removes the dependence of rover power and heating on sunlight[1]. Future rover missions may be able to operate at all times of day, virtually doubling the time efficiency of rover missions. Nuclear generators might also enable a rover to survive the cold and dark of an extended dust storm. Furthermore, waste heat from the nuclear source can be used directly to warm the rover; solar power must be converted to electricity to drive electric heaters. However, nuclear power may not completely solve the power management problem. Thermoelectric generators may not have sufficient output power to supply continuous locomotion or other high-power activities. Extended high-power activities might have to be supplemented with battery or solar power. A nuclear rover would still have to plan strategically to take advantage of quiescent periods of the day and night to charge the batteries in anticipation of high-power periods. Limited power and required charge cycles may actually prevent certain paths or greatly reduce daily range.

Thermal control will be another issue influenced by exposure to the sun. A nuclear powered rover, exposed to the waste heat from inefficient thermoelectric cycles, might be prone to overheating. In that case, a rover might adopt a motion strategy that occasionally seeks shadow to avert thermal buildup. If staying warm is the dominant thermal challenge, staying in sunlight could save power that would otherwise go toward heating rover electronics.

1.2.2 Lunar Polar Circumnavigation

The moon's South Pole Aitken Basin is a probable target for future rovers. Orbital missions over the past several years indicate a high probability of water ice trapped in permanently shadowed regions of the lunar poles, and hence present a strong scientific motivation for surface exploration (e.g. [13]). During summer months at the pole, the sun rises no higher than 1.5°, and from the point of view of an observer there would appear to skim over the complete horizon in the course of the moon's 29.5-day lunar month [23]. A combination of axial tilt and orbital eccentricity cause the Earth to inscribe a tilted elliptical path in the sky that rises to 6.7° above the horizon at its high point and falls to 6.7° below the horizon roughly two weeks later. The low sun and Earth elevation angles, combined with the South Pole’s rough terrain, are cause for widespread and highly varied sun and communications shadowing. Shadow patterns change continually with the moon's rotation and progress of the Earth/moon system about the sun.

A rover in this challenging environment could not survive without a path planner whose solutions maximize sun exposure and communications while satisfying operational constraints. Planning could discover paths that follow the course of sunlit regions to enable solar power and avoid extended exposure to the cold of lunar night. Such paths could also follow regions with direct line-of-sight to the Earth and relay spacecraft to allow high-rate imagery, teleoperated control and continual science data return. Mission objectives might force the planner to deviate from these zones of relative safety. Entering a region of permanent dark to look for signs of water ice would force the rover to abandon sunlight and to enter low-lying areas where communications might be occluded by surrounding terrain. A

1. Nuclear generators suffer from several disadvantages - launch approval for nuclear devices is extremely difficult, and generators are only made in a few different sizes which may be poorly matched to rover capabilities and demands.

mission planner would aid in timing this foray to maximize science data collection and rover contact while maintaining an adequate battery state-of-charge and maximizing the chance of survival.

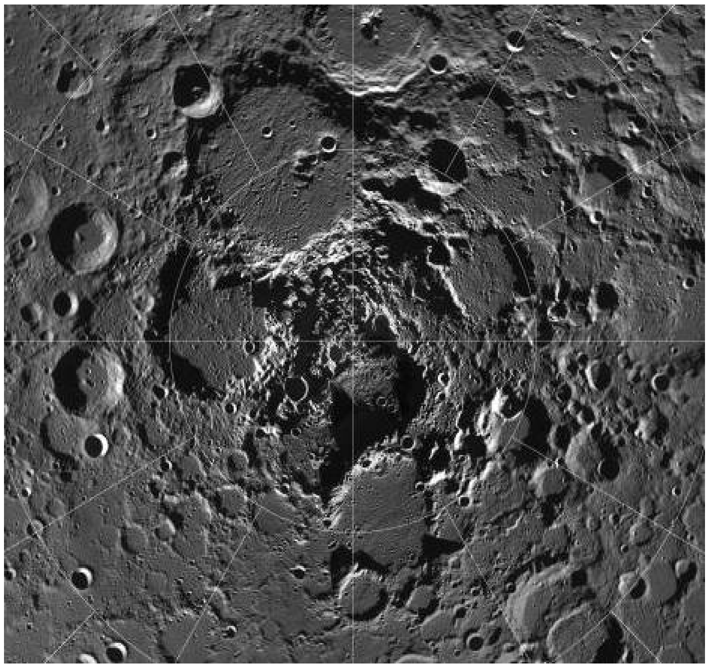


Figure 1-4: The Lunar North Pole. Future exploration missions may investigate permanently shadowed craters in search of water ice deposits. Such operations would require detailed traverse planning to anticipate terrain hazards, power availability, thermal transitions and areas of sunlight and shadow for science.

1.3 Mission-Directed Path Planning

As part of a broad effort towards planetary rover autonomy, this research introduces a new ideal for path-based reasoning based on the following five desirable attributes:

1.3.1 Over-the-Horizon Foresight

A critical task in achieving rover autonomy is automatic route planning between a landing site and operations sites. To date, path planning research for planetary rovers has focused on the problem of navigating locally through fields of rock obstacles en route to a global position goal, over tens of meters. In upcoming missions, long-distance and long-duration path planning will enable robots to travel between landing sites and operations sites. To target specific locations for scientific study, a robot must be able to traverse at least the size of the landing error ellipse, which could be tens of kilometers. To complement local path planning strategies, tailored for travel amongst rocks at or below the scale of the rover, path planning must utilize regional map data generated from orbit or during descent. Regional data will enable a rover to anticipate opportunities and hazards and to incorporate these predictions into path selection.

Large scale and duration introduce factors absent in local path planning, including navigation around and through large-scale terrain, significant changes in line-of-sight geometry, and time-varying lighting stemming from planetary motion. Long treks anticipated for future missions will demand that rover planning consider these issues.

1.3.2 Temporal Cognizance

Time is as important to path planning as it is to mission activity planning. Planetary motion defines the gross schedule for daylight, solar flux and opportunities for communications downlinks to Earth. A planner that considers the paths of the sun, Earth and orbiting relay spacecraft, and determines whether they are shadowed or visible at specific times will be better able to select routes that provide sufficient energy or enable communications at different times of day. A path planner must also ensure that paths obey time-dependent operational constraints. As examples, a science activity may only be successful under particular time-dependent lighting or thermal conditions; communications passes often require both geometric visibility and ground antenna availability. A path planner that operates in a mission context must use time efficiently and effectively, and tailor its paths to respect the operational constraints on other activities.

1.3.3 Resource Cognizance

Resource management is essential to rover self-sufficiency. Resources take many forms, from metric resources like battery energy and onboard memory, to unit resources like cameras, whose usage state is Boolean. The favored approach for rover resource management is through AI planning and scheduling (see [3] [10]). This approach proves effective in situations where path or orbit selection and resource management are independent. In surface vehicle operations, a tighter coupling of path and resource considerations offers distinct advantages over the traditional approach. Many resource expenses and gains for rovers are path-dependent and cannot be adequately considered outside a path planner. In the case of energy management, these include locomotion energy as a function of terrain and solar energy as a function of position, orientation and time. A rover must consider the effects of path choices on energy balance to determine which paths are feasible or optimal.

1.3.4 Uncertainty Robustness

It is essential that a rover planner be robust to uncertainty. Knowledge of the area of operations will be limited. Environment and rover models will be purposely coarse to ease the computational burden of considering long-distance and long duration paths. Rover behavior will be impossible to predict accurately. A robot that cannot adapt to unexpected events will, at best, be unable to operate for long durations because plans quickly become invalid, and at worst, may execute inappropriate or dangerous actions that could result in mission failure. At minimum, a robot should be able to perform quick re-planning when its state strays outside acceptable bounds of an earlier plan, or in response to new information about itself, the environment or goals. However, by anticipating the effects of uncertainty sources on the cost of future actions, a robot can select plans that avoid hazardous situations with sufficient, but not overly conservative margins.

1.3.5 Mission Directedness

Robot software should be designed to maximize the chances of achieving mission objectives. A path planner must balance a number of competing mission pressures, for example to maximize science data return, to maintain adequate battery levels, or to remain safe, all of which impact the mission outcome. A path planner with mission focus must represent all relevant activities - navigation, science, power generation to name a few - in a consistent framework. Navigation affects the timing of science activities, and could enable or prevent opportunities for battery charging. The location, timing and resource requirements for science activities impact navigation and may determine whether batteries require dedicated charge time. By reasoning about all significant activities, at the appropriate level of granularity, a path planner can correctly integrate route selection, timing, and energy management into a cohesive mission profile.

In considering operations over long distance and duration, it might be inappropriate to select a typical path metric such as distance or energy to evaluate plans, or alternatively, to select an arbitrary weighting to define the balance between several desired metrics. If the value of mission objectives is encoded in terms of reward, then the appropriate balance between these factors is achieved by maximizing the expected reward over the path. Suddenly, a single framework promotes the correct strategy in every situation. If a higher reward specifies the shortest path, the planner will seek it. If low batteries threaten rover survival, the planner will develop a course of action to charge to adequate levels as it proceeds to the goals. Finally, a rover planner would benefit from an ability to balance reward against risk. The expected return from a mission does not take into account the variance of reward. Avoiding undue risk may entail taking a route for which expected reward is lower, but for which the chances of failure are lower. A robot that can evaluate risk, and balance it against potential rewards could adjust its behavior according to mission preferences.

In combination, the above five attributes define an *ideal* for rover navigational autonomy - mission-directed path planning. Mission-directed path planning will enable a rover to autonomously achieve mission objectives, enforce operational constraints, and combat the effects of uncertainty under a single framework, and optimize plans in terms of probability of mission success. This thesis has developed an *initial* capability for mission-directed path planning embodying elements of each characteristic:

Over-the-Horizon Foresight: This research develops models and planning approaches that consider large-scale terrain, and execution approaches that integrate naturally with local navigation planners.

Temporal Cognizance: This research enables planning in an absolute time frame, and develops models and planning approaches that consider time-varying sunlight and solar power and absolute operational constraints.

Resource Cognizance: This research presents two approaches to optimizing path selection in terms of renewable stored energy.

Uncertainty Robustness: The research presents a strategy for fast re-planning in response to updates in vehicle state and localized changes in models of the environment.

Mission Directedness: This research integrates planning for navigation, temporal path planning, path-based resource management and satisfaction of constraints on mission activities.

1.4 Thesis Statement

This thesis asserts that mission-directed path planning achieves a significant, practical advance in planetary rover autonomy, and enables a new, challenging class of planetary surface rover missions.

1.5 Assumptions

Planning will occur at a spatial resolution at which the size of the vehicle and vehicle steering radii are insignificant.

Planning will consider scales and vehicle speeds at which dynamics are insignificant.

Planning will not solve the general planning and scheduling problem.

This research will not consider adversarial domains.

The research will use only deterministic models for planning.

Multi-goal planning will not solve or approximate solutions to the Traveling Salesman Problem.

The research will only consider optimal or resolution-optimal planning approaches.

1.6 Dissertation Roadmap

Having introduced the concepts of mission-directed path planning and established the assertions of this research, the remaining chapters answer the following questions:

- What techniques currently exist, and how do they fall short in solving this class of problems?
- What approach does this research take?
- How does the approach perform in laboratory examples?
- Is the technique useful in real-world problems?

Chapter 2 is an overview of current planning techniques relevant to path planning considering time, resources and constraints.

The work in this thesis relies heavily on incremental search strategies. Chapter 3 describes a previously-developed approach to incremental search, upon which this thesis builds.

Chapter 4 is the heart of the thesis. It further motivates and develops the ideas of mission-directed path planning, and introduces the approach that is the foundation of this work. The TEMPEST planner, one of the research contributions, is a representative mission-directed path planner. The second half of Chapter 4 presents experiments done in simulation that demonstrate the utility of TEMPEST and mission-directed path planning in general on space-relevant problems.

The highlights of this research are the demonstrations of TEMPEST in support of solar-powered robots in highly space-relevant terrestrial field trials. Chapter 5 illustrates how TEMPEST solved for plans that enabled two 24-hour, multi-kilometer traverses exhibiting a new large scale motion strategy for polar exploration called Sun-Synchronous Navigation. In a second field experiment, described in Chapter 6, TEMPEST generated plans to interleave long-distance routes with science activities for two solar-powered robots in support of robotic astrobiology.

Chapter 7 discusses the findings of this research, presents its principal contributions, and suggests several avenues of future research.

2. Related Work

The goal of this chapter is to prove the need for this work and its novelty relative to past approaches. The following sections examine past accomplishments in the fields of path planning and classical planning and scheduling and point out their shortcomings in the context of mission-level path planning.

2.1 Deterministic Path Planning

The objective of robot path planning is to determine a trajectory that achieves a goal state while avoiding contact with obstacles. Path planning typically occurs in the configuration space, whose dimensions each correspond to a degree of freedom in the robot. Motion sequences in the real world define trajectories in configuration space. Furthermore, obstacles in the real world map to regions in configuration space encoding the ranges of state variables over which motion is not legal. Path planning algorithms follow three basic approaches - cell decomposition, roadmap methods, and potential field techniques. Latombe [38] treats each in significant depth. A generalization of the problem assigns weights (non-uniformly) to all regions of the configuration space. The cost over a trajectory is the integration of the weights over the path. Obstacles have infinitely large weights, and free space has weight of zero. The planning problem then becomes to find the path that minimizes the cost between two points.

2.1.1 Cell Decomposition

Defining a regular grid or lattice that decomposes a space into regularly-spaced regions is formally called approximate cell decomposition. Grids are favored from a software implementation standpoint since they naturally map to array data structures and cell adjacency is implicitly encoded. Because a grid pattern only approximates the boundaries of free space, approximate cell decomposition techniques are incomplete. At the expense of time and memory, grids can be made arbitrarily fine, but where large regions of a space are homogeneous, grids are particularly memory inefficient. In many representations, motion in a regular grid is constrained to either the four-connected or eight-connected graph directly linking adjacent cells. As a result, planning in a grid cannot find the minimum-distance path between two points unless they lie along the directions of allowable motion.

Multi-resolution grids extend the regular grid approach to improve memory efficiency for spaces that contain large regions of free space and obstacles. In two dimensions, a space is initially divided using the coarsest possible square cells. Those cells that intersect both free space and obstacle space are subdivided into four equal cells each half the size of the original. The process continues until remaining cells sharing free and obstacle space are of the minimum allowable cell size and cannot be divided further. The decomposition can be represented as a quadtree, where parent nodes are large cells and child nodes are the smaller cells. The result is that large regions of free space are represented efficiently, and borders between free and obstacle space are represented finely. Szczerba and Chen developed a further refinement, the framed quadtree [68], which provides highest-resolution cells along the borders of all coarse cells to enable closer approximations to the shortest-distance path on a multi-resolution grid.

Used first by Chatila [8], exact cell decompositions divide a space into cells whose union is exactly the free space. The borders of the cells typically correspond to the borders of free space, or mark other properties of the obstacle shape. Because the cells match the free space, the technique is complete provided the search over the cell connectivity graph is complete.

None of the cell decomposition approaches were conceived with natural terrain path planning in mind. Though one could imagine a specialized decomposition for natural terrain, approximate decomposition techniques, especially regular grids, are attractive for their simplicity. Perhaps more importantly, existing data for natural terrain is almost universally stored in raster-patterned maps that form a regular grid.

2.1.2 Roadmap Approaches

Voronoi diagrams are examples of topological mappings, called retractions, that map the free configuration space to a one-dimensional subset [48]. The retraction is a “roadmap” graph that can be searched for a feasible or optimal path. Voronoi diagrams have the advantage of maintaining the greatest distance between a robot following the diagram and obstacles. Extensions to higher than two dimensions are available but are far more complicated to define. It is not clear how the Voronoi approach might be used in weighted spaces or in the context of natural terrain.

In two-dimensional spaces with polygonal obstacles, the visibility graph approach introduced by Nilsson [46] defines line segments between all obstacle vertex pairs that are “visible” to each other (i.e. whose inter-vertex line segments cross only free space). Adding the start and goal state as extra vertices, path planning becomes searching the resulting graph for the sequence of edges that connects the start and goal. The graph is guaranteed to contain the minimum-distance path between points. The approach is simple for two-dimensional planning around obstacles with distinct vertices, but is poorly defined in spaces with rounded obstacles, and may not lead to efficient paths in higher-dimensional spaces.

One might also envision a specialized roadmap for travel on natural terrain.

2.1.3 Potential Fields

Potential field algorithms [33] plan by superposing an attractive field centered at a global goal with repulsive fields surrounding obstacles. A "plan" simply follows the steepest resulting gradient to the goal. The approach is extremely simple to encode, but under the greedy, steepest descent strategy, is vulnerable to local minima, and hence is incomplete.

2.2 Randomized Path Planning

Randomized search techniques were developed to address motion planning for robots with many degrees of freedom (e.g. snake robots), and hence that have high-dimensional configuration spaces. Randomized algorithms trade a systematic, optimal approach for exploring a space for a stochastic approach that enables rapid exploration of very large state spaces. While deterministic, systematic search is complete and sometimes optimal, randomized techniques have probabilistic bounds on time to reach a solution, and are generally not optimal. However, for high-dimensional problems where finding any feasible path is sufficient, randomized approaches are very effective.

2.2.1 Rapidly Exploring Random Trees

LaValle's rapidly-exploring random trees (RRT's) are incrementally constructed search trees that attempt to rapidly and uniformly search the state space [36]. Their benefits include guaranteed convergence to a uniform coverage of any non-convex space. The basic technique seeds a graph with a node at the start state. A new point is randomly selected from a uniform distribution over the search space. In a greedy fashion, the node in the graph nearest to the random point is then expanded in the direction of the point to a non-collision state, becoming a new node in the graph. The process continues until the graph connects the start state to the goal state. Despite the greedy approach, the algorithm avoids local minima in high dimensions. Kuffner and LaValle present a follow-on algorithm, RRT-Connect [35], which grows an RRT from both the start and goal and attempts to greedily connect them together. Though non-optimal and incomplete, the algorithms are probabilistically complete and are efficient in practice for spaces with high dimensions (e.g. 10 or more).

2.3 Temporal Path Planning

Temporal path planning seeks time-optimal solutions, typically in a dynamic environment. The bulk of research in this area has been directed towards robotic manipulation and simple vehicle control, including dynamics and moving obstacles.

Bobrow *et al.* [5] approach the time-optimal path problem for a manipulator by dividing the problem into two steps. First, the path of the manipulator end effector is planned, using kinematic constraints, to avoid obstacles in the environment. Second, an algorithm solves for the optimal velocity profile, consisting of alternating segments of maximum acceleration and deceleration (bang-bang control). The algorithm considers kinematics, full manipulator dynamics, and joint torque limits that are arbitrary functions of the manipulator state. The dynamics and constraints result in a maximum velocity profile over the path. The optimal velocity profile follows the maximum velocity as closely as possible using the bang-bang approach. Though the approach provides a convenient way to decompose the optimal control problem, the assumed initial path may have a strong influence on the resulting maximum allowable velocity profile, and hence on the optimal solution. A global consideration of path and velocity might lead to significant improvements in performance. The Shiller and Gwo paper mentioned earlier [58] presents the analog to [5] for a robotic vehicle on natural terrain. However, unlike in the manipulation paper, the authors initially solve for the top several minimum-distance paths, then determine the optimal velocity profiles for each. This acknowledges that the initial path selection is not straightforward and that the minimum-distance path is not necessarily the optimal-time path.

Fraichard [17] introduces the concept of state-time space to solve temporal path plans for a 2 DOF non-holonomic vehicle. As in the work by Bobrow *et al.* [5], a path is selected in free space that avoids all static obstacles. Defining a parameter of path length along the path, the author defines a state-time space comprising dimensions in the parameter, its first derivative and time. Dynamic obstacles can be represented as volumes in this new space. A second step discretizes the space and evaluates canonical trajectories in terms of dynamic and collision constraints to derive approximately time-optimal trajectories. The obvious shortfall is that the path is pre-selected with no consideration of vehicle or obstacle dynamics. However, this thesis draws upon the notions of a state-time space or configuration-time space.

Fiorini and Shiller [16] define the concept of velocity obstacles to generate paths that avoid obstacles using the velocity space. The approach avoids path integration to predict robot position, but rather defines regions of the velocity space that predict imminent collisions. By superimposing regions of velocity that are accessible based on dynamic constraints, the technique is able to generate velocity profiles that avoid the obstacles with physically achievable maneuvers. Though not guaranteed to be optimal, this approach cleverly removes some of the complication of representations that consider the state-time space. It performs well for obstacle avoidance, but is not suited to the problem of planning through dynamically-varying cost regions.

By operating at a level in which rover dynamics can be ignored, the research in this thesis is able to avoid some of the complexity addressed in the above examples. However, time-optimality, and certainly time-dependence will remain important in many applications, and must be balanced against different competing factors, particularly finite resources or other performance constraints and mission objectives. All the above approaches take a hierarchical approach: the path is selected based only on avoiding static obstacles. The velocity profile is then selected to avoid the dynamic ones. This is seen as a substantial shortcoming. In the path-based resource management problem, path and timing must be considered simultaneously.

2.4 Resource Path Planning

In the realm of analyzing rover resource collection and usage, Shillcut [57] combines terrain, rover and ephemeris models to determine the evolution of solar array sun exposure and locomotion power for various coverage search patterns. The research rates the coverage patterns based on driving distance each requires as compared to solar energy collection each enables, as a function of time and rover orientation. Though the work stops short of planning paths based on this analysis, it provides motivation for integrated path planning and resource management.

Several other authors have considered planning to minimize the energy expense (work) over a terrain path [52][54][67]. These are described in more depth in Section 2.7.2.

2.5 Path Planning in Unknown Environments

Nourbakhsh and Genesereth propose an assumptive planning architecture to create limited conditional plans [47]. A complete conditional plan must consider all possible initial conditions, percepts and actions in a search for an optimal plan. As an alternative, the authors suggest a principled way to make simplifying assumptions about possible initial conditions, the effects of actions and the state of the robot given percepts, and yet to preserve plan correctness and goal-reachability in the face of wrong assumptions. In short, each function must return a proper subset of the actual states or action effects. Along with functions that make assumptions about initial state, actions and percepts, the architecture requires a function that can detect irreversible chains of actions (to prevent making an irreversible incorrect decision), and another function that detects false goals (to identify cases where actions must continue despite the sensors' indication of goal completion). The authors present an algorithm that interleaves planning and execution so that at each step, the robot executes the first action, senses its environment and re-plans the remainder. This reduces the requirements on the irreversible chain detector to look only one step ahead. The strategy was used in several successful robot control architectures in indoor environments, allowing the robots to act optimally in navigation tasks.

Stentz presents the D* algorithm, which is designed for global path planning in partially-known environments. It is a heuristic search algorithm, like A*, that visits the minimum number of states in finding an optimal solution, but generalizes to enable optimally efficient path repair in response to changing cost information [63][64][65]. Where A* is forced to plan from scratch if any state transition cost changes, D* determines which nodes in the graph are affected by the changes, and isolates the repairs to those nodes. The effect is a dramatic improvement, as many as two orders of magnitude, in speed over initial planning for cost map changes local to the robot. The algorithm has been used successfully in a number of real-world applications. An algorithm presented by Koenig and Likhachev [34], D* Lite, has more recently achieved even better performance than D* under a simpler design that more closely mimics A*.

The re-planning behaviors of D* and D* Lite are ideal for mission-directed path planning. However, neither enables global constraint satisfaction, nor are they optimized for use in state spaces of dimensionality greater than two.

A large body of research addresses the associated problem of planning under uncertainty, both for path planning and more generally. Though a treatment of uncertainty is not the focus of this thesis, notable path planning related works include those by LaValle and Hutchinson [37], Takeda, Facchinetti and Latombe [69], Wellman, Ford and Larson [76], Roy et al [55], Ferguson and Stentz [14] and Gonzalez and Stentz [21].

2.6 Path Planning Under Global Constraints

Standard heuristic search algorithms, like A*, assume the path metric can be specified in terms of a scalar-valued objective function that encodes cost over paths. However, in many applications, several disparate factors contribute to cost, promoting many to use a sum of weighted costs formulation. However, if any of the costs are non-linear, it is not immediately clear how to select weights in order to prioritize the factors to match the objectives of the search. A more natural approach is to specify constraints on these factors, and to plan paths that satisfy the constraints over the path. The following research is directed towards this approach.

Logan and Alechina [40] describe an extension of A* called A* with bounded costs, or ABC. ABC allows inequality constraints (i.e. cost < X), equality constraints (feature = Y), and optimality constraints (cost < optimum + ε) on a list of path costs that are tracked through each path transition. Using a user-supplied priority on these constraints, and an admissible heuristic on costs, ABC follows an algorithm very similar to A* that is both complete and optimal[1]. A path p is preferred to path p' if it has the same or better values for all of its cost functions. Distinct from A*, if both paths terminate in the same state, then p will *dominate* p', and hence will always represent an equal or better solution than will p'. In such cases, the dominated states can be legally removed from consideration while preserving optimality and completeness. All undominated paths must be tracked through the search. In under-constrained problems, the algorithm will add slack in prioritized constraints. The advantage of this approach is a clear flexibility in specifying the requirements for path under a framework that is natural for many applications. However, it is not clear from the paper how the computational and storage complexity increases with numbers and types of constraints.

Stentz also addresses the global path constraint problem, and adds a rapid re-planning feature to address applications in environments with incomplete or uncertain information [66]. The CD* algorithm uses a weighting factor to balance an optimality cost and a feasibility (global constraint) cost. CD* performs a binary search on the weighting factor to find the path that is optimal in the weight space under the global constraint. At each depth in the tree CD* uses D* [65] to plan an optimal path that satisfies the feasibility cost to within a weight factor error that decreases by a fac-

1. In fact, in the degenerate case where ABC has a single optimization constraint, its operation is identical to that of A*

tor of two, on average, with each iteration. In re-planning, CD* re-uses the graphs created by D* in the initial binary search to find a new weight, taking advantage of the fast re-planning characteristics of D*. Changing the weight of a D* graph is equivalent to re-specifying all the costs in the graph, eliminating the benefits of D*. Often, very few of the weights change with updates to the costs, and so CD* is often far more efficient than planning from scratch with A*. At the worst case, the weight must be adjusted at the root of the binary tree, forcing CD* to plan the entire problem from scratch. The author expects that an extension to multiple constraints will add a factor of the binary search depth D in run-time complexity.

Finally, under an approach similar to D* itself, Stentz and Tompkins offer an algorithm called Incremental Search Engine (Chapter 3) that provides optimal planning and re-planning under multiple global constraints in high-dimensional state spaces, yet with greater predictability in performance than CD*. As with D*, ISE runs in a fashion very similar to A* for initial planning, and re-plans efficiently because it determines which portions of the search space are affected by new information and limits the recomputation to those portions. ISE is space efficient through the use of dynamic state generation, state dominance and pruning within resolution equivalence classes.

Modeling constraints is essential for managing resources. The ABC algorithm is attractive because it appears to allow a very flexible means of specifying constraints on a path planning problem, and achieving combinations of optimality, constraints and slack. However, because it mimics the A* algorithm, changes to underlying models will often entail full re-plans over the state space. In this regard, CD* is an interesting alternative. However, its current restriction to 2-D search and single constraints may limit its value for the high dimensionality of this problem. *Of the three approaches, ISE is the most relevant to this research because it combines optimal and complete search, compatibility with multi-dimensional state spaces and fast re-planning.* As a result, it was adopted for use in this research. Chapter 3 describes ISE in full detail and demonstrates its performance under varying conditions.

2.7 Applied Path Planning: Natural Terrain

Roboticists have applied path planning to the problem of finding safe and optimal paths through rough terrain, both for military and space applications. The majority of research has focused on the local terrain problem, though a few efforts have developed approaches for global, long-distance planning. Note that the primary research issue in these examples is to measure and represent natural terrain, and that finding the plan itself invokes a more general path search algorithm.

2.7.1 Local Path Planning

In the local problem, knowledge of the terrain is typically limited to that derived from rover sensor measurements, and hence is limited to areas immediately around the rover path. Rover navigation schemes often interleave sensing, planning and execution at high frequencies to re-assess local rocks in the context of finding a minimum-cost path to a global goal position. Furthermore, local methods often represent the rover's volume (as opposed to representing the rover as a point), kinematic constraints, and occasionally, dynamic constraints.

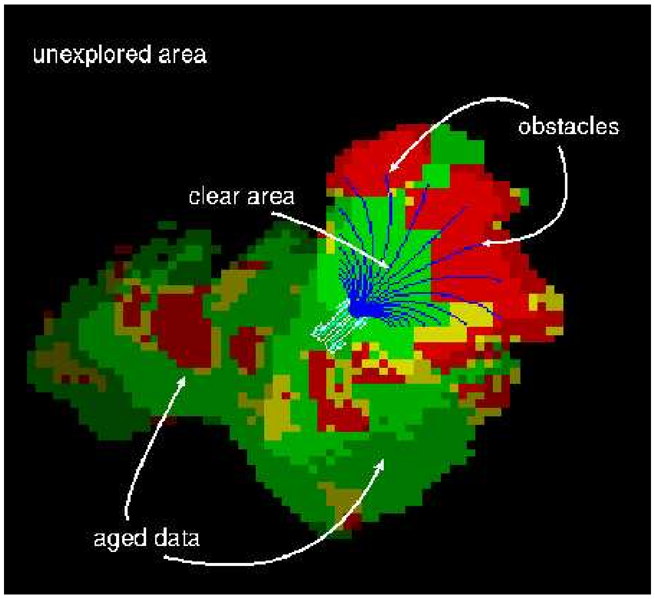


Figure 2-1: An example of terrain classification and arc evaluation from the CMU Mars Autonomy software by Singh *et al.* [59]

Given the recent pull for rover technology applicable to Mars exploration, research has sought to achieve the baseline required performance for the NASA Mars Exploration Rover mission [11]: local navigational autonomy for up to 100 meters over rough terrain.

RoverBug, as described by Laubach and Burdick [39], utilizes local tangent graphs to construct minimum-distance paths about obstacles detected by rover sensors with limited range and fields-of-view. Using a dual strategy of "motion-to-goal" and "boundary following", the algorithm has successfully demonstrated path planning and execution aboard the JPL Rocky 7 Mars rover prototype.

Howard and colleagues present a fuzzy logic terrain classification methodology for mobile robot navigation [24]. Using vision sensing, the approach loosely quantifies several terrain characteristics - terrain roughness, slope, discontinuity, and hardness - using fuzzy variables. The method composes the resulting vector of fuzzy variables and applies a set of rules to classify the traversability of the terrain. The authors claim a greater stability in classifying terrain traversability using noisy sensors than with analytical approaches.

Morphin, a local path planner presented originally by Simmons *et al.*, and in refined form by Singh *et al.* [59], uses data from stereo vision or other 3D sensors to track local terrain traversability. A terrain classifier uses stereo range data to derive a "goodness" measure comprising ground plane orientation and surface roughness, and measurement certainty, over grid cells approximately the size of the robot (see Figure 2-1). Morphin merges goodness maps over time, by taking the weighted average of the goodness values, scaled by the certainty measures. It de-values older data by reducing their certainty measures as a function of distance traveled since the measurements. It projects a predetermined set of discrete drive arcs over this local map and computes the goodness and certainty of each arc. Arcs intersecting obstacles or entering entirely unknown regions are vetoed. The overall value of an arc is computed as the multiplication of its goodness and certainty. This local scheme has been successfully demonstrated, in conjunction with a D*-based global path planner, on a CMU ATRV robot [59], on the CMU Hyperion rover [78], and for tens of kilometers on a Honda ATV automated for the DARPA PerceptOR program [32]. A modified version of Morphin, called GESTALT, provides local navigational autonomy for the NASA Mars Exploration Rovers [19].

Both RoverBug and the Morphin-based approaches coarsely categorize terrain for traversability, but generally avoid the roughest terrain. Other approaches are working to plan paths over rough terrain, and consider a greater number of factors including vehicle kinematics and dynamics. LAAS has developed a method for planning over rough terrain in terms of rover kinematic constraints, described in Hait and Simeon [22]. Assuming an *a priori* model of the terrain, the method estimates how a rover will sit on terrain by “placing” the rover over the entire discretized terrain map. Given estimated wheel contact, the software checks for violations in joint limits and under-body clearance. Those cells where placement yields no violations are used for planning using terrain roughness and distance metrics.

Kinematics often impose constraints that must be considered in planning local paths. With automated fork trucks in mind, Kelly and Nagy developed an approach for generating non-holonomic trajectories reactively [31]. Rather than solve the general non-holonomic path planning problem, their software operates on polynomial spiral trajectories. The spiral primitive has the advantage of roughly spanning the space of feasible steering controls while being described by only two parameters. Optimal control laws on the polynomial coefficient parameters transform to a non-linear programming problem which can be solved very quickly on the spiral primitives. Their results display an ability to generate and evaluate trajectories in under one millisecond. This approach was successfully adapted for use in rough terrain navigation for the DARPA PerceptOR program [32].

Researchers at MIT are developing physics-based motion planners for planetary rovers that consider vehicle statics, kinematics and dynamics and soil mechanical properties [25]. This approach hopes to provide a physics-grounded means of planning paths in rough terrain such as steep embankments where simple heuristic classifications of terrain might fail. Research by Urmson is integrating randomized kinodynamic projections of the rover state to path planning to enable travel over rough terrain that requires vehicle inertia to succeed [72].

Wellington and Stentz present an adaptive technique for local navigation through vegetated terrain [75]. In the presence of tall grass, bushes and other vegetation on terrain in front of a vehicle, laser scanning and radar data cannot easily deduce the height of the weight bearing surface. Some measurements reflect the full height of the vegetation, while others may represent mid-level branches and possibly the ground. The approach demonstrates both off-line and on-line learning to determine the height of the ground from this ambiguous data. With an estimated map of the ground height derived from learned experience, software samples from the range of possible future control actions, taking into account steering dynamics, to project a set of possible drive arcs. The software “places” a kinematic model of the vehicle onto the terrain model at regular intervals along the arcs to test for violations of roll and pitch limits, mechanism limits, and underbody ground contact (similar to [22]). A simple planner selects the non-violating arc that produces the lowest cost to a given goal.

All the planners mentioned above are solutions to varying forms of the local path planning problem. Their sensor-based models of rover-scale terrain are of limited use in over-the-horizon navigation or in sensing, modeling or identifying large-scale terrain. Though some might address obstacles of arbitrary size, perhaps only [25] and [72] considers large-scale terrain in a more general framework of slopes and their effect on static stability, traction and locomotion energy, as a function of the rover. None of the local planners address time-varying parameters, for example sunlight, in path selection. Furthermore, these approaches attempt to reduce energy expense by minimizing path length and avoiding rough terrain, but ignore the non-monotonicity introduced by energy collection and the global constraint imposed by limits in storage capacity. Finally, each of these methods only partially address uncertainty. For example, the D*-based method interleaves sensing, re-planning and action at a high frequency to restrict execution of plans to the immediate proximity of the rover, where data is most certain and complete. The LAAS approach anticipates uncertainty in rover state by ensuring that configurations over a corridor of positions and orientations about the path also satisfy kinematic constraints. However, in avoiding probabilistic or worst-case uncertainty growth over the path, its plan may not adequately prepare for the range of possible outcomes.

2.7.2 Global Path Planning

Long-distance path planning is characterized by a greater *a priori* knowledge of terrain over vast distances than for local planning, albeit at very low resolution. At large scales with low-resolution terrain data, it becomes impractical to consider obstacles at the scale of the rover, but becomes more convenient to model travel more abstractly. Terrain map grid cells define regions of homogeneous terrain properties, and the robot is reduced to a point object travelling from cell to cell.

Shiller and Gwo [58] present a vehicle path planning problem in which path and vehicle speed are optimized. The approach ignores vehicle size relative to terrain, but does model basic vehicle steering kinematics and dynamics. Based on the approach of Bobrow *et al.* [5], the technique splits the time-optimal problem into two by first determining the top several minimum distance paths, and then optimizing each of their velocity profiles using kinematics, dynamics and dynamic constraints. An obvious disadvantage with this sequential approach is that the minimum-distance path is not necessarily the time-optimal path. Aside from the approach used to optimize the path, an interesting feature of the approach is the use of B-spline patches and B-splines to model the terrain and paths, repectively. These continuous parametric functions enable a smooth integration of trajectories, but prevent optimization using gradient descent since the parameters to be optimized are the control points for the spline curves.

Another approach, by Pai and Reissell, represents terrain at multiple resolutions using wavelets, and plans paths hierarchically from the coarsest to finest maps [49]. The authors show that when applying wavelet filters in natural terrain, the wavelet coefficients in smooth areas shrink quickly at greater filtering levels (i.e. at higher resolutions), while in rough terrain, coefficients persist through many iterations. Therefore, in smoother areas, the lower-resolution wavelet levels match the original data well, while in rough areas, the coarse levels model the original terrain with significant error. Under this observation, the authors use an error measure at coarse terrain levels, a quantity that falls directly from the wavelet filtering operation, as a metric for terrain smoothness. Using the smooth terrain metric, the planning algorithm seeks smoothness-optimal paths at the lowest resolution, and then advances to higher resolutions in an "anytime" manner to refine the original path. The authors also examine various means of computing cumulative path cost, and derive a generalized worst cost function that heuristically prefers safest paths. Under experiments using digital elevation models of actual terrain, the algorithm appears to select the best paths through terrain. The approach here is novel, and potentially very useful in this work as a means of terrain analysis. However, it takes a heuristic rather than model-based approach to cost calculations, so is difficult to use in conjunction with power estimation that is required of an energy-cognizant path planner.

As mentioned in Section 2.1.1, grid-based representations suffer from memory inefficiency and the restriction of motion to paths between cell centers (or grid vertices). Another body of work plans paths on terrain approximated by polygonal regions with homogenous properties. This representation efficiently maps homogeneous regions and allows motion to occur in arbitrary directions. Richbourg et al. [52] introduce a technique for finding optimal paths, in terms of energy expense, where costs within the regions are isotropic - where the path cost per unit distance does not vary with path heading. They show that optimal paths consist of straight line segments across regions and direction changes between regions governed by Snell's Law from optics literature. This restriction of paths to Snell's Law trajectories, combined with several pruning rules, enables an efficient search of the region-wise continuous space for optimal paths. Rowe and Ross [54] extend this work, again for minimum energy expense paths, to travel through regions of anisotropic costs - where legal directions on sloping terrain are defined by maximum vehicle power limitations, sideslope tipover, and downhill braking. They prove that optimal paths can only cross these regions in one of four ways and, as in [52], that direction changes between regions must follow Snell's Law. Sun and Reif [67] apply an efficient discretization to the polygonal region boundaries, using Steiner points, and a more efficient search algorithm, to display good search performance on maps of natural terrain.

In another extension of the work by Richbourg, et al., Rowe and Lewis [53] describe a method for defining paths for both land and airborne vehicles to either minimize or maximize visibility with respect to fixed observers (e.g. representing beneficial resources or enemy observers) while minimizing path cost in terms of an energy metric. The planner divides the free space into convex visibility regions, defined by visibility to each observer, and assumes each of these volumes to be homogeneous in terms of both energy and visibility costs. Assuming linear costs, the authors apply the Snell's Law model to path propagation and minimum-cost path angle transitions between regions. Though interesting, the method appears more directed towards flight, and in its quest to abide by Snell's Law, uses overly simplistic, linear cost models. However, to its credit, the method does raise the issue of effective space decomposition, and addresses the balance between visibility and energy costs.

Several robots have used the D* algorithm [63],[64],[65] in conjunction with the Morphin local planner (Section 2.7.1) to drive autonomously on rough terrain over long distances. Local maps derived from stereo camera data are also used to populate a cost map in D* based upon estimated terrain traversability. D* produces an optimal path to a global goal position, given all available information. Unlike Morphin, which assigns infinite cost to unknown regions, D* planners have historically been designed to assign zero cost to areas with no existing data. Typically these no-data regions are at the field-of-view limits of the cameras, and will ultimately contain data as the rover gets closer. As new data is introduced, D* efficiently re-plans by repairing only those regions affected by the changes. Hence the optimal global path updates with each stereo image set, typically several times per second. The local obstacle avoidance and global planner each vote with their respective arcs; the higher-value arc is sent as a command to the rover controller. A relative weight factor allows operators to tune the behavior of the robot. Weights emphasizing obstacle avoidance will steer further from obstacles, but may avoid legal paths through narrow passages. Conversely, boosting the weight of the global path planner improves the likelihood of finding paths, but also increases the chances of collision with obstacles. The combination of local and global path planning has proven itself in many natural terrain rover experiments [59], [78], [32].

Of recent popular interest is the DARPA Grand Challenge automated racing contest. In 2004, teams developed vehicles and software systems to drive autonomously 143 miles in under 10 hours on a combination of rough roads and off-road conditions. Though no team completed the course, Urmson *et al*. describe Sandstorm, the system that drove further and faster than all other competing vehicles [73]. Teams were supplied the race route only 3 hours prior to the start of the race, in the form of GPS waypoints spaced an average of 89 meters apart, maximum speed limits, and maximum allowable corridors of travel. Sandstorm path planning was a mixed-initiative system that combined automated route generation on a regular grid at 1 meter resolution to follow the GPS points while staying within corridor bounds and minimizing travel on unknown and poor road conditions, subsequent automated route vectorization to compress and simplify the plan representation, and human inspection, correction and smoothing of paths using various graphical interface tools. Automated planning used coarse road classifications, as well as distance from the center of the drive corridor, as a basis for driving cost. It appears that a velocity profile was selected in a separate step after route planning. Onboard GPS and a scanned laser enabled Sandstorm to follow the course 7.4 miles at speeds up to 36 miles per hour. This system clearly did not freely select its global route, but rather connected pre-selected waypoints and established a speed profile that would enable completion of the route within the race deadline.

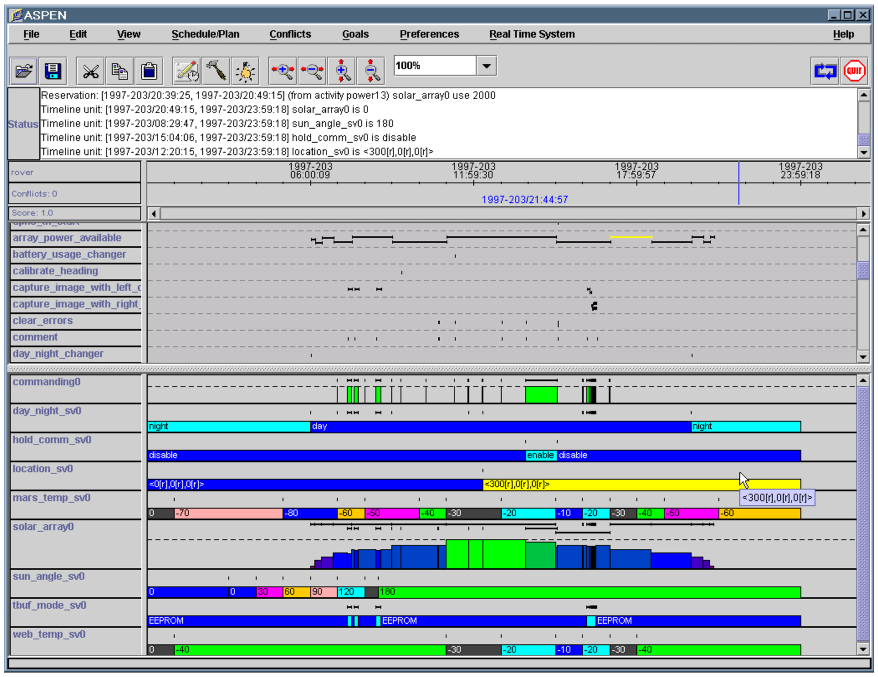
Despite their applicability to long-distance path planning, none of these approaches considers time-dependent costs. For example, though the representation of terrain using homogeneous cost regions and Snell's Law in [52][54][67] is attractive, it is not clear how to extend the model when cost regions are time-varying. Furthermore, the cases that seek minimum-energy paths ignore sources of incoming energy, such as solar energy. They minimize resource costs, but none models renewable resources where a balance must be achieved between input, output and storage. As a result, none can predict how a finite battery capacity might prevent certain paths, or how resource collection might extend the set of feasible paths. However, in terms of operational utility, the efficient re-planning provided by D*-based planners is clearly advantageous.

2.8 Planning and Scheduling

In an entirely different vein of research, the artificial intelligence community has evolved two related fields - *planning* and *scheduling* - whose overarching goal is to create feasible sequences of activities that start from a set of pre-existing conditions and achieve a desired set of goal conditions. Planning decomposes high-level goals into atomic-level activities that collectively achieve the goals. There may be many alternate ways to break tasks into subtasks, and activities may have complex interrelationships that prevent loose ordering. Scheduling takes a set of activities and determines an ordering that respects constraints and resource demands to accomplish an overall task.

Automated activity planning and scheduling software has been successfully deployed on spacecraft and prototype planetary rovers. The Remote Agent Experiment demonstrated automated planning and scheduling onboard the Deep Space 1 spacecraft [4], [3]. The planner/scheduler software (PS) accessed a database of long-term mission objectives and planned concurrent activity schedules over multi-day planning horizons. The PS derived planning problems by taking goals relevant to the planning horizon and projecting initial spacecraft conditions to the anticipated plan execution time. Beginning with an incomplete plan, the PS searched over plan space, adding constraints and subgoals where necessary and reordering activities, to create full partially-ordered plans. The PS also maintained onboard state predictions, for example resource levels, under the same model as activities in primitives collectively termed tokens.

ASPEN was developed to automate planning and scheduling for spacecraft and rovers [10]. The software uses a most-committed, local, heuristic iterative repair approach to decompose high-level mission goals into concurrent action sequences that respect operational and resource constraints. Because the search is not systematic, there is no guarantee that all combinations of actions will be searched exhaustively, or that disadvantageous sequences will not be examined more than once. However, ASPEN achieves planning, scheduling and resource management at far less computational expense than for more flexible, exhaustive approaches. The ASPEN system and the derivative CASPER system each produced coordinated activity schedules, based on science and engineering team requests, for the JPL Rocky 7 rover (see Figure 2-2). The activity planners repair plans in response to changing goals and other unexpected events. Rover activity scheduling considers resources and environment effects (e.g. day/night cycle, sun angle), but demonstrates only loose coupling to path planning. ASPEN calls a path planner that is ignorant of absolute time and resources to estimate the duration of small traverses. ASPEN's primary focus is on conflict resolution through event rescheduling or reordering. Furthermore, ASPEN and CASPER essentially react to evolving initial state and updated mission goals. Neither has the ability to plan in anticipation of a range of possible outcomes, and so cannot build in contingency branches to plans.



Reservation: [1997-203/20:39:25, 1997-203/20:49:15] (from activity power13) solar_array0 use 2000
Timeline unit: [1997-203/20:49:15, 1997-203/23:59:18] solar_array0 is 0
Timeline unit: [1997-203/08:29:47, 1997-203/23:59:18] sun_angle_sv0 is 180
Timeline unit: [1997-203/15:04:06, 1997-203/23:59:18] hold_comm_sv0 is disable
Timeline unit: [1997-203/12:20:15, 1997-203/23:59:18] location_sv0 is <300[r],0[r],0[r]>

Figure 2-2: A screen shot from the ASPEN scheduler. Rover activities are shown as ticks in the middle rows. The colored bars represent mission-relevant parameters, for example solar array current, sun elevation, and surface temperature, over the course of one Mars sol

2.8.1 Contingency Planning

Contingency planning is addressed by Bresina and Washington under the Contingent Rover Language (CRL) and the propagation of expected utility distributions [6]. Combining the action condition structure of classical planning and conditional rewards characteristic of decision-theoretic planning, the authors present a means of efficiently evaluating contingent branches in an existing plan. The method specifies probability distributions on execution times for all events, and defines local rewards conditioned on action success or failure. Valid action execution is defined under start, wait-for, maintain and end conditions, which include requirements on initial available resources. The method assumes knowledge of resource availability as a function of time. Given an existing plan with conditional branches, the algorithm first forward-propagates distributions of possible action start times, given action duration distributions and conditional requirements on actions, to establish temporal bounds on outcomes. Then, it backward-propagates distributions of utility from the branch endpoints, restricting computation to the time distributions. The resulting utility distributions specify the expected value of executing each branch, conditioned on start time. The advantage under this approach is that the utility distributions remain valid regardless of the temporal outcome of actions, as long as the resource bound model also remains valid. Though this approach pre-supposes a profile of resource availability, it estimates contingent branch utility factoring in whether actions can be legally executed. An extension of contingent planning that reasoned about uncertainty in activity time and resource consumption, using CRL and the Pico planner (based on EUROPA [28]), was used to successfully demonstrate multi-target single-cycle instrument placement on the K9 rover [50].

2.9 Summary

Mission-directed path planning seeks to solve for paths in a context historically addressed in the AI planning and scheduling literature. Path planning is most naturally solved in a state space formulation; the planning and scheduling community has almost universally used spaces of possible plans. Planning and scheduling software deals very effectively with temporal and resource constraint problems, but has not addressed problems where spatial dimensions are so critical, as in path planning. Specifically, planning and scheduling has been most successful at *satisfying* resource constraints, but has not typically sought to *optimize* resource usage. Optimization of state parameters is the domain of the state space. Meanwhile, the path planning community has addressed temporal planning in a hierarchical fashion that prevents globally-optimal solutions, and has limited its treatment of the resource management problem to resources that are consumed monotonically over a path.

As demonstrated by ASPEN and CASPER, AI planning and scheduling software can call a path planner as a subroutine to handle mobile phases of a larger activity plan. In scenarios where traverses are relatively short-distance and duration, this type of hierarchy might suffice. But in situations where traverse activities consume a large share of the time and resources in a mission plan, the system cannot tolerate a path planner that is naive to time, resources and constraints. *Addressing the combined problem of path planning, temporal planning and resource planning is a central contribution of this thesis.*

This thesis elects to follow the path planning paradigm. The following is a brief list of problems in mission-directed path planning that are largely absent in the robot path planning literature but have been addressed in this thesis:

- Efficient, optimal planning and re-planning, under global constraints, in a state space of greater than two dimensions.
- Simultaneous, global consideration of highly-coupled route, time and resource variables.
- Incorporation of time-varying anisotropic costs and constraints.
- Capability to reason about, constrain and optimize non-monotonic (expendable and replenishable) resource variables.

- Representation of actions, either coupled with navigation actions or distinct, and both mobile and stationary, that enable resource management or achieve non-navigational mission goals.

The Incremental Search Engine by Stentz and Tompkins embodies several attributes that are highly desirable or even essential to solve the mission-directed path planning problem. ISE offers the optimality guarantees and re-planning efficiency of D*, which has proven itself in many robot applications. It is also more space and time efficient in addressing higher-dimensional state spaces than other existing algorithms. This is critical if we expect to consider time and resources. Furthermore, ISE enforces global constraints on paths, for example bounds on battery energy. The next chapter describes ISE in far greater detail as background for presenting a practical solution to mission-directed path planning.

3. Incremental Search

This chapter describes one of the foundations for this work - an algorithm developed by Stentz and Tompkins, called Incremental Search Engine (ISE), that satisfies many of the demands of mission-directed path planning. ISE provides several enhancements over previous algorithms that are critical in mission-directed path planning. ISE is a generalization of the D* algorithm (see Section 2.5 or [63][65]) that retains the planning and re-planning efficiency of Focussed D* [63], but enforces global constraints and operates efficiently in state spaces with greater than two dimensions. Mission-directed path planning is the first and most exhaustive application of ISE.

ISE is a backwards-chaining algorithm that starts its search from one or more goal states, and finds an optimal path that "ends" at the specified start state. Like A* and D*, ISE maintains a list of states to "expand" to propagate the search to new states. These states, beginning with the goals, are prioritized for expansion according to an objective function which accumulates the cost from each state to the goal, and a heuristic function which estimates the cost from the start state to each of the states. State expansion generates all possible backward "arcs", representing actions in the planning domain, from the final state to several initial states originating the actions. The new states are placed on the list, and prioritized according to their own value. The optimal path emerges upon expanding the start state. Further machinery within the prioritized state list enables ISE to efficiently repair path solutions when the transition arcs change during a path execution.

The following sections describe the salient features of the algorithm. Section 3.1 describes how ISE encodes application domains. Section 3.2 explains several efficiency mechanisms which speed search and minimize state proliferation. Section 3.3 provides an overview of the search process (Appendix 1 describes the ISE algorithm in detail). Throughout these sections, it should become clear that ISE is a general incremental search algorithm that can be applied to a wide range of problems. A user must define state space variables and application-specific functions of state to compute state transitions between states, heuristics, and state priorities, and other quantities. Section 3.4 experimentally demonstrates how ISE performance varies as these parameters change.

3.1 States, Transitions and Cost

A*, D* and other incremental graph search algorithms are best suited for two-dimensional grid planning, or in domains where states are abstract nodes in a graph. In factored spaces of greater than two dimensions, existing algorithms fail to provide a simple means of encoding state spaces. ISE allows a natural representation of state space in terms of discrete independent and dependent parameters.

3.1.1 Independent and Dependent State Parameters

ISE allows two types of state variables - independent parameters (IPARMS) and dependent parameters (DPARMS). IPARMS are the foundation of the ISE state space and search and are essential to defining a problem domain. Changes in IPARMS are independent of other state parameters. A domain may optionally include DPARMS, parameters whose values change as a function of changes in IPARMS or other DPARMS. DPARMS for a state are stored in a set corresponding to the state's IPARMS. An ISE state combines the IPARMS and DPARMS of the domain.

For example, a robot control problem might entail planning a path through a two-dimensional office area, where obstacles are time-varying. The state space might define two independent spatial parameters X and Y that are positions in a regular grid, and a dependent temporal dimension T, such that $\Delta T = f(\Delta X, \Delta Y)$. The ISE state is an (x, y, t) tuple. ISE would represent this domain by encoding a regular grid of sets, each corresponding to an *x*-*y* cell, that store values of *t*. Each *t* value would represent a different state (x, y, t).

3.1.2 State Transitions

Arcs are transitions between two ISE states. ISE uses two user-defined arc transition functions, one backwards (β) and one forwards (Φ), to model state transitions in a domain. Given a state space S and action space A, the arc transition functions define deterministic mappings $S \times A \to S$ that obey the Markov assumption, that is, the resulting state is a function only of the action and the originating state. Given a state and a choice of action, the transition arc functions define the change in IPARMS and the corresponding change in DPARMS. The backward function β takes a final state and produces the initial state from which the action was executed. In many cases, the forward function Φ takes an initial state and produces the state resulting from the execution of a given action. In such cases, β and Φ are perfect inverses ($\beta = \Phi^{-1}$).

The forward transition function Φ is not always single-valued. ISE state variables can be constrained to minimum or maximum values. For example, a state variable b might represent battery charge which can never rise above the battery capacity b_{max}. Function β could legally take the form $b_{initial} = min(b_{final} - i\Delta t, b_{max})$, where i and Δt are the the charging current during the action and the action duration, repectively. Now, suppose there is an action A where

i is negative (signifying a battery discharge in the forward-time direction). If $b_{final} \geq b_{max} - i\Delta t$, then $b_{initial} = b_{max}$. Clearly, then, the forward transition function Φ for action A cannot unambiguously predict b_{final} from $b_{initial} = b_{max}$. Instead, Φ must predict the *range of legal states* that are possible from an initial state. Using the above example, Φ could take the form $b_{final} = b_{initial} + i\Delta t$ for $b_{initial} < b_{max}$, and $b_{final} \in [b_{initial} + i\Delta t, b_{max}]$ for $b_{initial} = b_{max}$.

Arc parameters, or APARMS, govern the effects of transitions. Each IPARMS combination has APARMS that help define the transitions into or out of the states with those IPARMS. Most importantly, APARMS are parameters that are allowed to change during plan execution. As discussed in upcoming sections, re-planning enables ISE to efficiently repair a plan whose underlying APARMS have changed.

3.1.3 Local Constraints

Local constraints are a generalization of obstacles in the path planning literature. The satisfaction or violation of local constraints depends solely on the specific state and the action executed to or from that state; it does not depend on state or action history. Obstacles in traditional path planning are local constraints on locomotion actions. Local constraints extend obstacles by limiting arbitrary actions over any range of states in the state space. Through the application-specific (user-defined) arc transition functions β and Φ, ISE permits or rejects arcs into states that violate local constraints. Rejected states cannot become part of any ISE path solution.

In the running example, the office robot would be constrained to avoid people and closed doors, as defined by local constraints at specific (x, y, t) regions of the state space. The office environment might further restrict its operation to certain hours of day, a purely temporal local constraint.

3.1.4 Global Constraints

In contrast to local constraints, which operate on state parameters, global constraints operate on parameters that integrate over the history of state or actions. Examples include path distance, duration, and capacity limitations on consumable or replenishable resources. ISE enforces global constraints in one of two ways. ISE stores global constraint parameters through auxiliary variables stored alongside DPARMS state parameters or, if the parameter is to be optimized, in the objective function (see Sections 3.1.6 and 3.1.7). If stored alongside the DPARMS parameters, the global constraint parameters are evaluated inside the β state transition function. When a violation occurs, the β function rejects the arc. Alternatively, if the parameter is stored in the objective function, ISE cannot reject the arc, but assigns a very high cost to the arc to reduce its chances of entering into any optimal solution. Under this second approach, if no cheaper legal path exists, ISE will yield the least expensive illegal path.

What are the implications of local and global constraint parameters? The addition of global constraint parameters does not add dimensions to the state space. However, under the Markov assumption, aside from rejecting arcs, *global constraint parameters cannot legally have any effect on DPARMS or cost for state transitions*. Furthermore, *ISE can only guarantee planning completeness in the variables comprising the state space*. If a parameter is not a member of the state, ISE cannot explore trajectories that vary the parameter to find feasible path solutions. Any global constraint can be transformed into a local constraint by promoting its parameter to a full DPARMS state parameter. The parameter can then affect state transitions and costs for legal state transitions, but at the computational and memory expense of an added dimension in the state space (see Section 3.4.2).

3.1.5 Resource Parameters

Because of its flexible treatment of state, constraints and cost, ISE is particularly well-suited to reasoning about resources. A resource is a quantity that is essential to achieving a goal, and yet is in limited supply. ISE represents resources as constraint-limited parameters. ISE can be represent resources as DPARMS state variables, limited by local constraints, or as global constraint parameters alongside DPARMS or in the objective function.

Resource parameters can be divided into two categories - monotonic and non-monotonic. Monotonic resource parameters monotonically increase or decrease over the course of a trajectory. Examples of monotonic resources include non-rechargeable battery energy or finite-lifetime components. More abstractly, if a path is constrained to be less than a given distance or duration, then remaining distance or duration can also be considered resources. Planning problems often involve renewable resources - parameters that can be expended and replenished over different arcs, and hence are non-monotonic. Rechargeable energy, thermal load, available computer memory or communications bandwidth are all examples of non-monotonic resources.

In ISE, because of the backwards-chaining order of search, the semantics of resource parameters are different than for typical parameters. As with all DPARMS and objective function parameters, values for resource parameters must be initialized for each goal prior to search. Semantically, it is useful to treat these values as minimum allowable resource levels at the completion of a path. For example, a parameter to represent duration remaining to the goal (as a way of constraining duration over a path) might be set to zero. In words, at the start of a path if a robot is given an upper bound on duration to reach the goal, it is legal in the worst case to exhaust the entire duration “resource”, but no more.

To explain the semantics of resource parameters at other points in a path, it is important to show how they evolve over a search. Goal DPARMS states and objective function values are used as a starting point for state expansions during the search. The state transition function β generates the change in state for DPARMS parameters as well as the costs that are reflected in the objective function. It *subtracts* quantities that would be *added* in the forward direction and vice versa. Returning to the example, the duration resource declines in the forward direction, and hence accumulates in backwards expansions. Figure 3-1 depicts the evolution of a monotonic duration resource over an ISE search. The plot's horizontal axis represents the sequence of states in a path in the direction of search, from the goal (extreme right) to the start (extreme left). The vertical axis represents duration, from zero to the maximum allowable duration D_{max}. The solid curve starts at the goal at zero duration remaining. Moving left from the goal, arcs successively add duration to the total. Under this model, it is important to understand that resource parameters do not represent the instantaneous value of the quantity, but rather the *minimum amount allowable to satisfy the goal conditions* under the current optimal arc sequence.

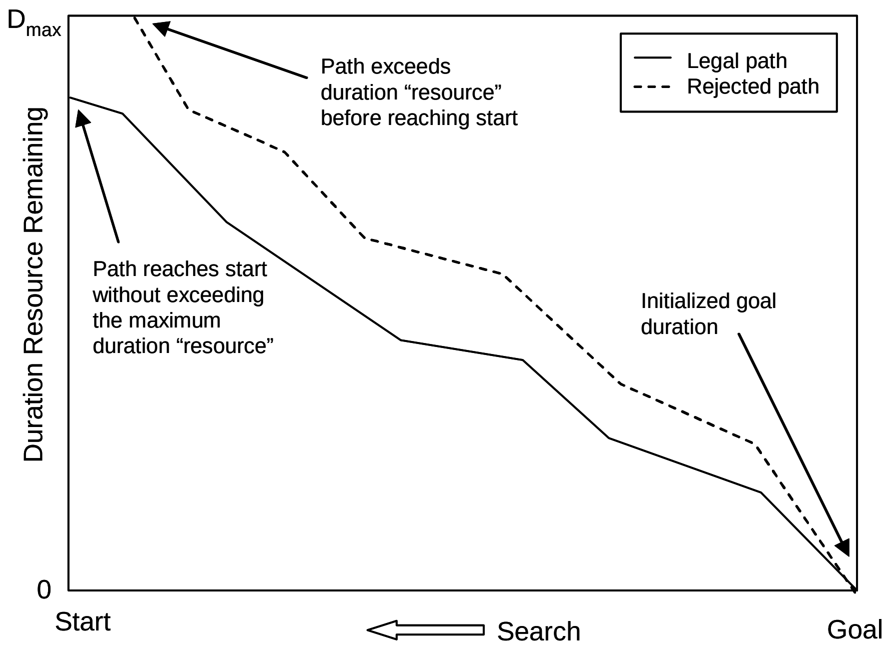


Figure 3-1: Plots of Duration, a Monotonic Resource Parameter, Over an ISE Search. The solid curve depicts a path that satisfies the global duration constraint, the dashed curve shows a duration profile that exceeds the maximum duration before reaching the start. Steeper slopes indicate slower progress to the goal.

To constrain a monotonic resource, the ISE user must design β or the objective function to reject arcs, either through outright rejection or via high path cost, that exceed the minimum or maximum parameter value. The example duration constraint rejects arcs that force the duration parameter above D_{max} (see the dashed curve in Figure 3-1); any such arc would force the robot to start with a greater duration allocation to reach the goal with no less than zero upon reaching the goal. The arc rejection mechanism is easily defined in either the state transition function β or within the objective function, as described in Sections 3.1.3, 3.1.4 and 3.1.6. Feasible paths arrive at the start state with margin remaining with respect to the maximum allocation (see the solid curve in Figure 3-1).

Non-monotonic resources are not as intuitive as their monotonic counterparts. Non-monotonic resources can go either up or down over any arc. Therefore, parameters that have finite ranges typically require upper *and* lower limits. Take, for example, a memory resource, where some arcs correspond to net data being stored, and others, net data being removed. Positive increments in memory in the forward direction (erase actions) appear as reductions in backwards search, providing greater margin with respect to the memory capacity. However, successive reduction arcs could drive the memory parameter to zero. Herein lies the subtlety: zero in the memory scale does not indicate that ISE should reject the arc. Instead, zero indicates that the goal could be reached from the current state, *even starting with no available memory.* ISE must prevent the parameter from dropping below the minimum - there is no meaning to a resource with "less-than-empty" conditions. However, the parameter remains legal, and can remain at zero or increase above zero in later state expansions.

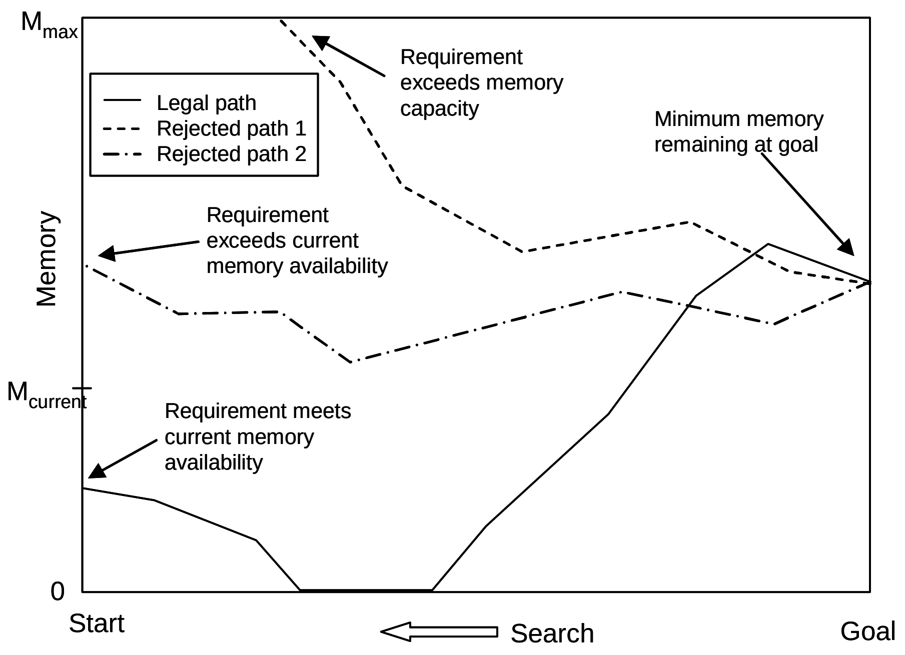


Figure 3-2: Computer Memory, a Non-Monotonic Resource Parameter, Over an ISE Search. The three curves depict different outcomes of the search: a path that requires more than the maximum available memory (Rejected path 1), a path that exceeds the memory available at the start (Rejected path 2), and a path that meets both the maximum and initial memory requirements (Legal path).

Figure 3-2 illustrates the memory example. Similar to Figure 3-1, the horizontal axis represents a sequence of states from the goal (extreme right) to the start (extreme left). The vertical axis represents remaining unused memory, and ranges from zero remaining memory to M_{max}, the full capacity of the storage device. All the example curves start at some allowable memory level (possibly non-zero). Observe that each curve displays non-monotonic behavior. The curve labeled “Rejected path 1” behaves very similar to the rejected path in Figure 3-1. It ultimately exceeds the maximum capacity of the storage device, meaning that the particular series of arcs taken by the path required there to be more memory than available to meet the goal conditions. “Rejected path 2” is also rejected, but for different reasons. In that case, the path never requires more than the storage capacity, but requires more than is available at the start. Perhaps all the data in memory is critical and cannot be deleted or moved out of memory. A solution that required deletion of this data would be illegal. This is rejected on the basis of feasibility, and is alluded to in Section 3.3.1. Finally, the “Legal path” illustrates how a resource curve can be reduced to zero and remain valid. The portion of the curve that dips to zero (from right to left) indicates that the arcs move more data out of memory than in. The flat part of the curve at zero memory indicates that the capacity for data removal during those arcs is sufficient to allow a full memory at that point in the path, and still meet the goal conditions downstream. Furthermore, the starting memory requirement falls below the amount of memory available at the start.

To represent non-monotonic resource parameters using DPARMS, the ISE user must design transition function β to reject states that exceed one extremum, and saturate the values that would otherwise exceed the other. A resource parameter r represents a minimum allowable level, and has upper bound r_{max} and lower bound r_{min}. For successive states r_i in a backwards search, where $\Delta r(a)$ is the change in resource level for a given arc a from a given state s in the *backward* direction, the next resource level could be given by:

$$r_{i+1} = max(r_i + \Delta r(s, a), r_{min}) \qquad 3\text{-}1$$

and would reject $r_{i+1} > r_{max}$. The implementation of non-monotonic resources in the objective function is not as simple, and requires a special formulation, described in Section 3.1.7.

3.1.6 Path Cost

As with A* and D*, ISE combines an objective function and a heuristic function to encode the cost of a candidate path. ISE searches in a backwards-chaining order, from one or more goal states to a start state. Backwards arc transitions between a state X and a second state Y result in a positive cost defined by the arc cost function $c(X, Y)$. If Y does not have a forward arc to X, then $c(X, Y)$ is undefined. The cost to traverse a sequence of states $S = \{x_1, x_2, \ldots, x_n\}$, such that $x_{i+1} = \beta(x_i)$, is the sum of the arc costs incurred in backwards transitions along the sequence.

$$\sum_{i=1}^{n-1} c(x_i, x_{i+1}) \qquad 3\text{-}2$$

The *estimate* of the cost of a path from an initial state R, through a state X, to a goal G is based on two functions, $h(X)$ and $g(X, R)$. Given a sequence of states S from a goal G to state X:

$$S = \{x_1, x_2, \ldots, x_n\} \text{ with } x_1 = G \text{ and } x_n = X \qquad 3\text{-}3$$

function $h(X)$ is the objective function that returns the cost accrued over the sequence of n arcs, and can be viewed as an estimate of the optimal cost from the goal to the state.

$$h(X) = \sum_{i=1}^{n} c(x_i, x_{i+1}) \qquad 3\text{-}4$$

Since $h(X)$ is a summation over a path, it can represent a monotonically-increasing global constraint parameter. An application can simultaneously optimize the parameter and limit its expansion within the same objective function. In the case where it violates a constraint, the cost for the arc $c(x_i, x_{i+1})$, and hence $h(X)$ itself, is set to a very large number that approximates infinity.

Function $g(X, R)$ is a focussing heuristic that estimates the cost from the state X to the state R. The heuristic function must be admissible - it must never over-estimate the cost between two states. A good heuristic approximates the actual cost as closely as possible without ever exceeding it. Both functions must be monotonic - $h(X)$ must monotonically non-decrease with number of steps from the goal, and $g(X, R)$ must monotonically non-decrease with number of steps from state R. The sum $f(X, R) = h(X) + g(X, R)$ is the estimated total cost between a goal G and a state R, through state X.

3.1.7 Non-Monotonic Path Cost

In some cases, it is desirable to optimize a path in terms of a non-monotonic resource. As an extension of standard cost functions like distance, duration or resource expense, ISE enables a user to encode *composite objective functions* that involve more than one parameter. One use for this mechanism is to regulate the search for an optimal solution in terms of a non-monotonic resource.

Non-monotonic parameters cannot be optimized directly using the incremental search approach. Heuristic search prioritizes its effort on the estimated lowest-cost solutions, based on the objective function. If the objective function is the sum of successive positive or negative resource costs, then, in general, the objective function can assume arbitrarily small values. In cases where negative cost arcs are consistently reachable, the search will never terminate - pursuing negative-cost arcs will yield continually lower objective function values with continually higher priority.

One solution is to add an additional term to the objective function which causes the sum to be monotonic. The new objective function takes the form:

$$h(X) = nK + \sum_{i=1}^{n} c(x_i, x_{i+1}) \tag{3-5}$$

The summation component is the same as in Equation 3-4, but may also include constraint mechanisms like in Equation 3-1. In the new term nK, n is the number of arcs in the sequence, and K is defined as follows. If c_{min} is the minimum cost taken over the entire state space S and action space A:

$$c_{min} = min_{\forall(x \in S, a \in A)} c(x, \beta(x, a)) \tag{3-6}$$

then K is defined by:

$$K = \begin{cases} |c_{min}| & \text{if } c_{min} < 0 \\ 0 & \text{if } c_{min} \geq 0 \end{cases} \tag{3-7}$$

In words, if negative costs exist in the set of possible transitions, K is the absolute value of the *most negative* cost over the state space. Therefore, the following always holds true:

$$K + c(x_i, x_j) \geq 0 \text{ for all } i, j \tag{3-8}$$

K exactly cancels out the largest possible negative cost, and outweighs all others. At every step, the objective function increments by the sum $K + c(x_i, x_j)$. Hence, the $h(X)$ defined in Equation 3-5 is monotonically non-decreasing. As with the original objective function, the summation of costs can also be governed by global constraint mechanisms like in Equation 3-1. Note that the new objective function does *not* optimize the sum of original costs $c(x_i, x_j)$. Because K is added at every step in the plan, the term nK is a measure of plan length. Therefore, the path solution that results from a search under a composite objective function is optimal in *combined terms* of the cost and number of plan steps.

This formulation is useful for optimizing non-monotonic resources in many applications: battery energy, fuel in a tank, data in memory, or thermal energy in an electronics component.

3.2 Efficiency Mechanisms

ISE employs three mechanisms - dynamic state generation, resolution equivalence pruning and state dominance - to counteract state proliferation during search. The following sections describe how each is used to manage the search memory and enable efficient path planning in greater than two dimensions.

3.2.1 Dynamic State Generation

While many search algorithms store all states in the space explicitly, ISE generates states only as they are encountered during the search, and deletes them if they become irrelevant. ISE employs the arc transition function on existing states to generate new states. ISE defines a state set for each combination of IPARMS values. Each state set stores states with common IPARMS but potentially differing DPARMS. A state set stores the DPARMS for each state possessing its particular IPARMS. The IPARMS are implicit in each state set.

At the beginning of search, only the goal states are stored in the sets. ISE generates states to search the space, but manages state proliferation by continually checking for state redundancy and irrelevance. Unnecessary states are deleted from the sets, thereby saving memory.

Returning to the earlier example, each pair of IPARMS values $(x \in X, y \in Y)$ defines an IPARMS set containing DPARMS values $(t \in T)$. A transition function might describe nine legal actions - a motion to each of eight neighbors, and a wait action that remains in the current (x,y) location for fixed duration. For each action, it defines $\Delta T = f(\Delta X, \Delta Y)$. APARMS for each IPARMS combination might include parameters to define robot speed as a function of position, or encode whether an obstacle prevents the action from occurring.

ISE deletes states as they become irrelevant for planning. The following two sections describe two mechanisms for state deletion - resolution pruning and state dominance.

3.2.2 Resolution Equivalence

In search over a discrete space, IPARMS dimensions can be represented at arbitrary resolutions, based only on requirements of the problem domain. Since changes in each axis can be considered independently, discretizing the space is a simple matter of independently discretizing each dimension to provide adequate resolution.

DPARMS change as functions of changes in independent or other dependent parameters. This leads to an important difference between the independent and dependent parameters. Transitions in IPARMS are integer multiples of the axis interval; dependent state transitions are, in general, real-valued. DPARMS must also be integer-valued. Real valued transitions must be rounded or truncated to the discrete intervals of the dimension. To avoid excessive rounding errors over many transitions, the resolution of DPARMS axes must be sufficiently high. However, in selecting an adequately high resolution, the number of discrete intervals in the dimension multiplies, leading to a vast increase in state space size and in time to search the space.

ISE enforces user-defined DPARMS equivalence classes to address this issue. In each IPARMS set, equivalence classes group DPARMS over ranges of values into coarse bins. The user selects the bin size to reflect an acceptable range over which DPARMS can be considered equivalent for the particular application. During a search, ISE uses the DPARMS equivalence classes to prune away states that are “resolution equivalent” but inferior in terms of cost. This dual resolution representation enables ISE to classify and prune away sufficiently similar but inferior states while avoiding the excessive rounding errors that would emerge in using only the coarse resolution.

Returning to the example, assume the office area is 100 x 100 meters. From the ISE perspective, the resolutions for the X and Y axes are arbitrary, but might be selected to be 1 meter. If transitions are only possible to the eight neighboring cells in the grid, the base resolution of T must be selected to approximate changes in time for both transitions to principal-direction neighbors and to neighbors along diagonals. For example, if the robot in this example moves at 0.1 meters per second, then the time to cross a grid cell in the principal and diagonal directions would be 10.0 and 14.1 seconds, respectively. A time parameter T with a 10-second resolution would not differentiate between the time costs in these directions; 10.0 and 14.1 both round to one 10-second time unit. However, a one-second resolution might be sufficient for the application.

Problematically, if the time parameter T spans one hour, the total number of states in the space is 36 million. However, by dividing the DPARMS into 60 bins of one minute each, the state space drops to 6×10^5 state bins while maintaining the rounding error for transitions at ± 0.5 seconds per step. The drawback is that resolution pruning reduces the accuracy with which path solutions match the initial state of a robot. Errors can be up to one half the resolution equivalence class. In this example, solutions could be mismatched to the current robot time by as much as 30 seconds.

3.2.3 State Dominance

In some applications, states obey a dominance relationship in which one state can be guaranteed to always produce a lower cost solution than another state. More formally, let S and Q be any two states in the graph and G be the goal state. Let $h(S, Q, G)$ be the cost of the optimal path starting at S, passing through Q and ending at G. Let $h(S, Q, G)$ be infinity if no such path exists. State A dominates B for a given G if:

$$h(S, A, G) < h(S, B, G) \qquad \text{3-9}$$

for at least one S, and

$$h(S, A, G) = h(S, B, G) \qquad \text{3-10}$$

for all remaining states S. If a state is dominated by another, then it can never be a member of an optimal path and can be removed from consideration.

ISE enables the user to define the specific conditions for dominance, if any, amongst states with common IPARMS but possibly different DPARMS.

3.3 Search

ISE searches to find optimal paths from a start state R to one or more goal states $g \in G$. The search is in backwards-chaining order, originating at the goal states. To search the space, ISE “expands” states from a prioritized list called the OPEN list. Initially, this list contains only the goal states. Given a final state, ISE uses the backwards arc transition function β to dynamically generate all possible initial states from which the final state is reachable. Through state expansion from the goals, ISE builds a directed graph whose nodes are states, and whose edges are the transition arcs. By definition, every node in the graph is a state from which one of the goals is reachable. The heuristic function focusses expansion on states predicted to reach the start state with minimum cost. The search terminates when the search expands states in the state set containing R that meet the criteria for an optimal solution.

In an initial search, ISE results are similar to those from A* [9]. With changes to transitions, either in arcs or objective function costs, ISE operates far more efficiently than A* in re-planning paths. The A* algorithm is forced to re-plan the entire space from scratch if any transition arcs change. ISE, like D*, can repair the search graph in the area of the changes, thereby drastically reducing the time for re-plans. Using incremental graph theory, ISE repairs the feasible set of solutions and the optimal path within it.

Sets whose arc parameters have changed have their member states placed back into the OPEN list. The effects of the changes propagate through the search graph under the same machinery used in initial search. The focussing heuristic limits repairs to states affecting the start state, making the algorithm much more time efficient than A*. The algorithm search direction is optimized for applications in which new arc information appears close to the robot's current state, for example from robot sensors. The search is backwards-chaining so that changes near the start state only affect "leaf" states at the very edge of the graph. If the search were in the forward direction, changes near the robot's state might have to propagate through the entire search graph, essentially as costly as a new search from scratch.

Another advantage of a backwards-chaining search is that state variables can be expressed in terms of goal state requirements. By initializing a goal state parameter to the minimum or maximum quantity allowed to achieve the goal, search state expansions propagate these requirements to other states. The semantics of the state parameter is the same at any step, including the first step in a plan solution - it reflects the minimum or maximum quantity allowable, from that state, to achieve the goal condition. This is a particularly advantageous operational tool. At any step in the plan, a robot can monitor and immediately determine, based on its current estimate of the parameter, whether the plan can feasibly achieve the goal. In contrast, a plan from a forward search is less insightful. Since state parameters in a forward search have their origin at the current robot state, they can only provide an estimate of the expected state of the robot at each step. The plan generated with a forward search does not provide a threshold for making operational decisions during execution. To generate a goal-referenced plan with a forward search would require multiple search iterations; it is not clear what iteration strategy would efficiently achieve the desired result.

3.3.1 Modes and Search Termination

ISE operates in one of two search modes, selected by the user. BESTPCOST mode finds the minimum cost path. BESTDPARMS mode finds the solution with the "best" start state DPARMS that falls below a maximum path cost. A more detailed description of these modes appears in Appendix 1.

In each case, search can terminate when the lowest value on the OPEN list equals or exceeds the path cost from the start state R. Since the cost of expanded states is monotonically non-decreasing, the OPEN list cannot possibly find a LOWER state that has a low enough path cost (effect of the objective function $h(X)$) and that is "close" enough to the start state to be able to reduce the path cost from R when it reaches it through subsequent expansion (effect of the heuristic function $h(X, R)$). Candidate start states must also meet a feasibility condition, typically to guarantee sufficient proximity of the candidate to the actual start state, or satisfaction of necessary start conditions.

3.3.2 Path Extraction

To extract the optimal path from the search graph, ISE uses the forwards arc transition function Φ to locate successive states in the sequence. Beginning with the start state, ISE calls Φ to dynamically generate candidate parent states in the plan. ISE compares each generated state to existing states with the same IPARMS set to find a state with matching DPARMS and path cost. The matching state is the next state in the optimal path, and is used to generate its candidate parents from the search. The process repeats until a goal state is reached.

3.4 Experimental Results

ISE can be adapted to a wide range of problems, and can be configured in multiple ways to solve the same problem. The configuration that one applies affects memory and computational performance, the quantity to be optimized and the thoroughness of the search. This section quantifies ISE performance under a variety of problem and configuration parameters, and explores the other effects stemming from the ISE configuration.

3.4.1 Test Domain

At the risk of losing generality, ISE experiments were conducted in a specific planning domain. The tests all solved for an optimal path between two positions. To emphasize ISE novel features, the problem sought optimal paths with respect to a renewable energy resource rather than the typical path length. The test domain is a two-dimensional grid of positions, with arc transitions defined from any grid position to each of its eight immediate neighbors (no stationary actions were possible). Arc transitions result in time and energy costs, randomly pre-generated in advance of each test for all cells in the map. Time costs were uniformly sampled over a strictly positive range, while energy costs were uniformly sampled over a range that enabled positive or negative costs.

The domain state space comprises four parameters: two position variables x and y, a time variable t, and the energy variable e. The two position variables define axes on a regular grid, and are ISE independent parameters (IPARMS). The time variable represents the time left to reach the goal, and is an ISE dependent parameter (DPARMS). The energy variable is the battery energy required to reach the minimum goal energy. As a renewable resource, energy is non-monotonic. Depending on the ISE mode used, as will be explained below, energy is represented in ISE in one of two ways. In both cases, e is constrained to be within the legal range of battery levels, from zero to some maximum value e_{max}

3.4.2 Comparison of Two Solution Approaches

This study contrasts two approaches for solving this resource management problem, summarized in Table 3-1. Approach 1 uses the BESTDPARMS mode (Section 3.3.1). In this approach, the energy variable e is represented as a second DPARMS parameter, yielding four dimensions for the search. Under BESTDPARMS, ISE finds the "best" DPARMS solution (lowest energy requirement to reach the goal, as defined by the function $better(X, Y)$) whose cost falls below a given maximum duration. Observe that even though the objective function for Approach 1 is duration, the BESTDPARMS uses it only to constrain the duration of solutions that have a minimal energy expense.

Approach 2 uses BESTPCOST mode (Section 3.3.1) and the composite objective function mechanism (Section 3.1.7), with energy as the non-monotonic resource parameter. Under BESTPCOST, ISE finds the minimum cost path in combined terms of energy and number of plan steps. The objective function represents the energy variable e as a global constraint parameter, and hence without using a DPARMS parameter. Therefore, the DPARMS e dimension used in Approach 1 is unnecessary in Approach 2. This yields a three dimensional search and a vast reduction in search space. Removing the energy dimension comes at a price, however, and is discussed later.

The objective function of Approach 2 formulates the resource cost $c(x_i, x_{i+1})$ of Equation 3-5 to enable a representation of the energy variable that is independent of the ISE IPARMS-DPARMS state. As mentioned earlier, e represents the energy required to reach the goal. This quantity can never be less than zero, since there is no meaning to negative stored energy. Furthermore, no feasible plan can require more than a full battery to achieve a goal. Both of these constraints must be reflected in the energy representation. If Δe_i is the change in energy predicted by the arc transition function $\beta(x_i)$ from state x_i, the resulting energy can be constrained to be above zero:

$$e_{i+1} = max(e_i + \Delta e_i, 0) \qquad \text{3-11}$$

Under Equation 3-11, the greatest change in energy possible in an arc is Δe_i. To limit the upper bound on energy, the resource cost term from Equation 3-5 then becomes:

$$c(x_i, x_{i+1}) = \begin{cases} e_{i+1} - e_i & \text{if } e_{i+1} \le e_{max} \\ \infty & \text{if } e_{i+1} > e_{max} \end{cases} \qquad \text{3-12}$$

If the resulting energy does not exceed the capacity of the battery, then the cost is simply the change in energy. If the resulting energy does exceed capacity, the path is infeasible and is given an infinite cost. For feasible trajectories, the summation of resource costs over a path correctly (Equation 3-5) represents the energy parameter e. The other term from Equation 3-5, path length cost K, is equal to the absolute value of the most negative energy cost over all possible transitions. The sum of the resource and path length costs is always non-negative, since the resource cost can never increment more than Δe_i. So, this adaptation of Equation 3-5 continues to guarantee a monotonic increase over the path, and also enables a full representation of the energy state parameter without the use of another state space dimension. Finally, the domain transition model does not depend on the energy variable to derive time or energy costs. Hence, an energy state DPARMS is redundant with the parameter in the objective function, and can be removed from the DPARMS of Approach 1.

Table 3-1: Summary of Experiment Approaches Using ISE Modes

Feature	Approach 1 (BESTDPARMS)	Approach 2 (BESTPCOST)
State Space	IPARMS: • x, y (position) cells DPARMS: • t (time to goal arrival) • e (battery energy)	IPARMS: • x, y (position) cells DPARMS: • t (time to goal arrival)
Transition Arcs	Arcs from cell to each of eight neighbors	
Costs	Randomly generated, uniformly distributed time and energy costs • Time costs: $U(5, 20)$ • Energy costs: $U(-35, 40)$	
Start and Goal Specification	Start position, energy Single goal position, time, energy	
Planning Details	ISE Mode: BESTDPARMS Objective function: Δt Better: if $e_1 < e_2$ Dominates: if $class(t_1) = class(t_2)$ and $e_1 < e_2$	ISE Mode: BESTPCOST Objective function: $nK + \sum_{i=1}^{n} \Delta e_i$ Better: if $t_1 < t_2$ Dominates: never

The two approaches differ in substantial ways. Most obviously, they operate on state spaces of different dimensionality. In terms of time and memory, the 3-D Approach 2 was expected to have a notable advantage over the 4-D Approach 1.

The approaches also seek optimal energy paths in totally different ways. As explained in Section 3.1.7, both approaches must either limit plan length or promote shorter plans to avoid the problems of heuristic search for non-monotonic costs. Approach 1 (4-D BESTDPARMS) seeks an optimal energy solution indirectly, since it uses a duration-based objective function as the mechanism for limiting plan duration. The objective function tracks the duration of candidate paths, and prevents paths that exceed a maximum duration. This prevents the search from lingering at states that provide negative costs. Under this limit, ISE uses state equivalence class pruning and state dominance to remove less desirable states. Within time-energy DPARMS classes, the $better(X, Y)$ function favors states that have *lower required energy.* Within *time* DPARMS classes (but across *energy* DPARMS classes), the $dominates(X, Y)$ function also favors states with lower required energy. The optimal solution in Approach 1 is one that falls below the given maximum path duration, and whose initial state is "best".

Approach 2 (3-D BESTPCOST) contains energy directly in its objective function. The plan length cost in the composite objective function provides incentive to avoid lingering at regions of negative cost. The objective function applies a path length penalty K at every step in the path. This contrasts with Approach 1, which does not penalize extra steps as long as the plan duration is shorter than the given maximum. At any step in the Approach 2 search, the plan length cost K can potentially cancel out the resource cost Δe. Consequently, there is no admissible heuristic to focus the search. Finally, in Approach 2, with energy removed from the DPARMS, no state dominance relationship exists. So, despite the advantage of having fewer dimensions to search, Approach 2 has fewer other mechanisms to help focus the search and to prune unnecessary states.

Approach 2 has some other disadvantages. Any arc transition must be a function of state (IPARMS and DPARMS) and action alone. Since the energy parameter e is not in the DPARMS, costs in Approach 2 cannot depend on energy. In this particular domain, there are no costs that depend on energy, but in general this could be limiting. For example, it would be impossible to model a battery whose discharge rate is a function of its state of charge in the Approach 2 formulation.

More importantly, Approach 2 is incomplete (in the planning sense) with respect to energy. Reasoning about state, defined by position and time, does not consider energy. In the event that one branch of a search does not yield feasible paths, it cannot backtrack to different regions of the energy space, since it is not represented in the IPARMS-DPARMS state. Paths that are eliminated due to high cost cannot be regenerated. In Approach 1, states that are dominated (and deleted) can be regenerated if it is discovered that the originally dominant states do not lead to a feasible solution. In practice, the incompleteness of Approach 2 does not interfere with finding paths in a wide range of situations.

Having introduced the example planning domain and the two candidate ISE approaches, the remaining sections assess the performance of ISE in response to various parameter changes. The tests to determine performance scaling with map size, DPARMS class resolution and branching factor (Sections 3.4.3 - 3.4.5) and to compare planning and re-planning (Section 3.4.6) all used Approach 2 (3-D BESTPCOST mode). Sections 3.4.7 and 3.4.8 compare the Approaches in terms of performance and solution quality. All tests were run on a Pentium 4, 2.99 GHz processor, with 1 GB of RAM.

3.4.3 Scaling With Map Size or Start-Goal Separation

Map size refers to the number of IPARMS levels - the IPARMS dimensions of the 2-D spatial portion of the space. The experiment varied the size of square maps (x and y axes 50, 100, 150 and 200 cells each), with start and goal positions at opposite map corners. Tests measured the time required to solve the planning problem under increasingly large map sizes and start-goal separation, using Approach 2 described above. Twenty tests were run for each map size, each on a different independently and pseudo-randomly-generated cost map. Figure 3-3 plots the ISE initial planning duration as a function of map size and confirms the intuition that a larger map size results in greater planning time. For each doubling of map dimension (quadrupling the number of IPARMS sets), the planning time increases roughly by a factor of 10. Computation should, in theory, be $O(N \log N)$ in the size of the search space. However, this study does not examine increasing the IPARMS *resolution* but rather increasing the *span* of the IPARMS.

On the surface, doubling the span of IPARMS would be appear to quadruple the state space. However, the DPARMS time dimension is open ended; the DPARMS time dimension grows to contain all the states expanded in the search. By expanding the size of the map, the distance between the start and goal increases, as does the duration of plans considered by a search. The size of the DPARMS time dimension is coupled to the distance between the start and goal position. An increase in the IPARMS dimensions actually increases the DPARMS span proportionally. The extra multiplicative effect seen in planning computation in Figure 3-3 may be explained by this coupling.

3.4.4 Scaling With Resolution

An ISE user can vary the resolution for DPARMS equivalence classes. Recall that DPARMS equivalence classes group states that are considered sufficiently similar for resolution equivalence pruning (Section 3.2.2) and state dominance (Section 3.2.3) at each step in the search. They also define the resolution at which goal states can be specified, and the accuracy with which planning can match start state DPARMS parameters to actual initial conditions.

Tests in this study varied the resolution of the time variable DPARMS for various map sizes. As with the map size study, each combination of map size and resolution was tested 20 times, using 20 independently, randomly-generated cost maps. Figure 3-3 shows the performance benefit of decreasing resolution in the DPARMS. Given base time units, the study examined DPARMS equivalence class resolutions of 30, 60, 120 and 240 units per bin. Each curve in the plot corresponds to one of the resolutions. Each map size results in a different mean plan duration (the time the plan requires to travel from the start to the goal). Therefore, bin size represents a different fraction of the entire time DPARMS axis depending on the separation between goal and start. The number of 30 unit bins required to span the mean duration of plans was 27.7 for the 50-by-50 map, 62.8 for the 100-by-100 map, and 85.3 for the 150-by-150 map. The search will tend to populate approximately the number of time DPARMS bins required to span the solution. As shown in the plot, by halving the resolution at a given map size, the planning time decreases by a factor of between 2.4 and 2.7.

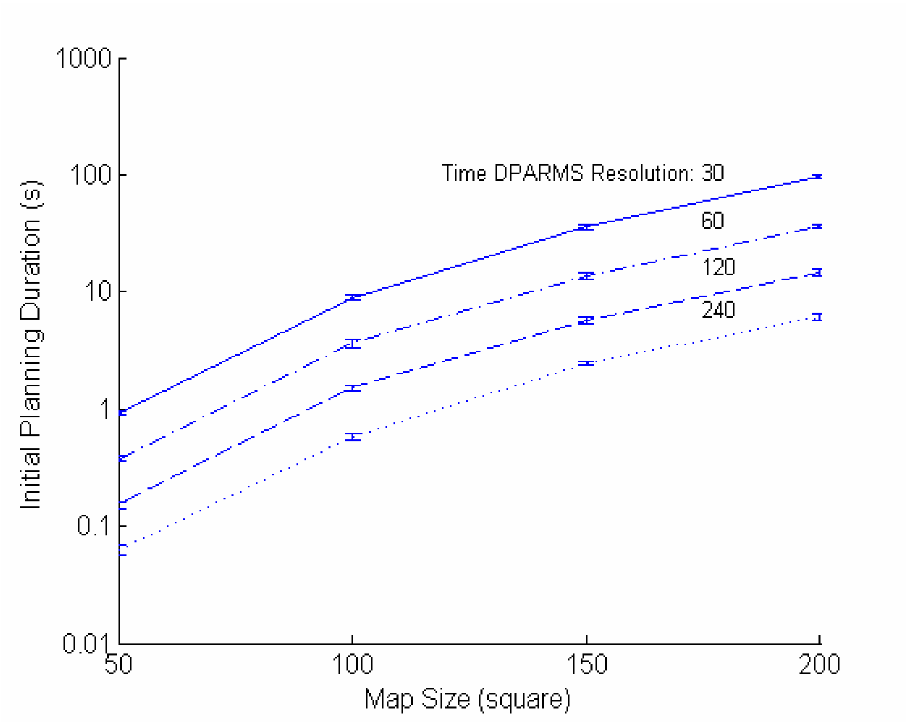


Figure 3-3: Approach 2 (3-D BESTPCOST) Performance for Initial Planning Time vs. Map Size and Time DPARMS Resolution

3.4.5 Scaling with Branching Factor

The number of available transition arcs from each IPARMS state is application-specific. This number equates to a branching factor in the search graph that affects both performance and path cost. For Approach 2 (3-D BESTP-COST) planning, experiments recorded the time required to solve for a plan under two transition arc models - the original, eight-neighbor arc set, and a reduced, four-neighbor arc set in which arcs were restricted to horizontal and vertical motions. In both cases, the start and goal positions were on opposite corners of a square map. Twenty experiments were run for each combination of map size, DPARMS time resolution and branching factor.

In a standard state representation, with a fine resolution in the dependent variable, the branching factor determines the exponential rate of state proliferation. A larger branching factor would lead to exponentially more states. Figure 3-4 demonstrates the power of state resolution pruning, with the eight-connected planning shown in solid lines and the four-connected planning shown in dotted lines. Somewhat counter-intuitively, Figure 3-4 shows that the eight-connected state space requires nearly the same time as the four-connected case. For ISE, branching factor does not substantially affect the time complexity of the search, because at each step, state resolution pruning discards states that are deemed inferior in each DPARMS state resolution bin. At each expansion step, the number of states in each IPARMS set remains roughly constant over the search, regardless of the branching factor. The number of states to expand depends more on the number of DPARMS equivalence bins are spanned by the search. ISE prevents the state explosion that would have occurred in a standard search, and yet represents dependent state variables at a fine resolution.

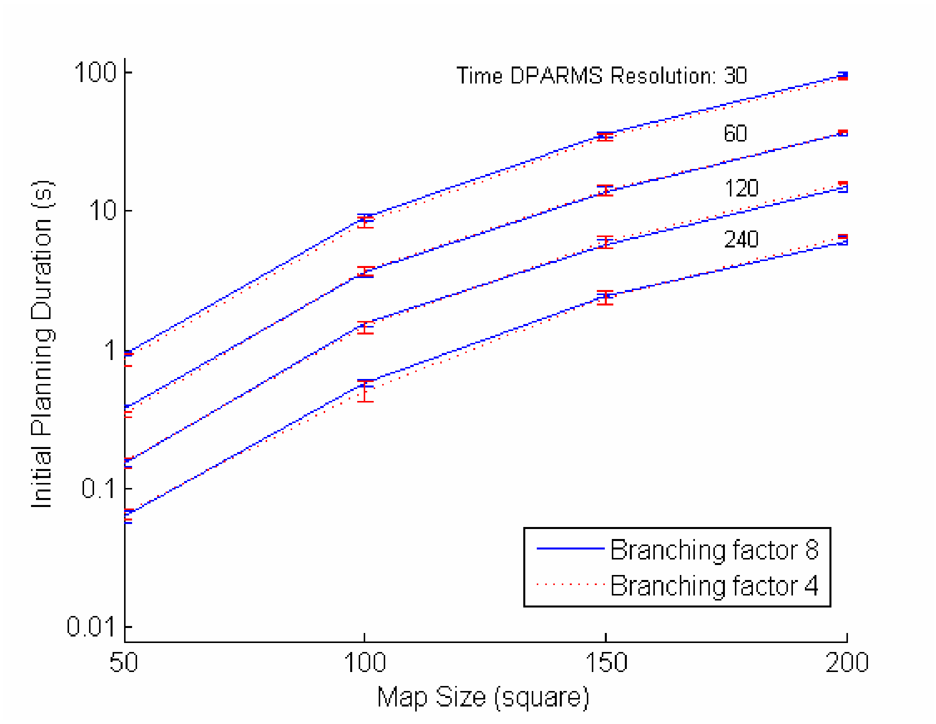


Figure 3-4: Time for Four and Eight Connected State Spaces for Approach 2 (BESTPCOST).

3.4.6 Re-Planning Performance

ISE re-planning was designed to efficiently repair plans in response to changes in arc costs near the start state. A set of tests compared the time required to plan from scratch versus re-planning in response to changes on an existing cost map. Cost map updates were restricted to four modifications within 4 x 4 cell windows for various map sizes and DPARMS resolutions (30, 60, 120 and 240 time units). In each test, planning was first conducted on a base cost map. The modifications were made to the costs on the original map, and re-planning repaired the initial ISE graph. The time required for the re-plan was compared against the time required to plan from scratch on the modified cost map.

Figure 3-5 compares the time to re-plan in comparison to planning from scratch, in response to cost updates near the start position. All cases show improvement over planning from scratch, especially at lower resolution. For larger map sizes, re-planning enables a factor of better than 100 speedup relative to initial planning. For smaller maps, modifications affect a greater portion of the entire map. The work of re-planning comes closer to replicating the work of initial planning, explaining the reduced benefits shown in the left side of the figure. Lower resolution in DPARMS axes reduces the influence of changes in the cost map on other DPARMS equivalence classes, thereby reducing the work to repair the graph.

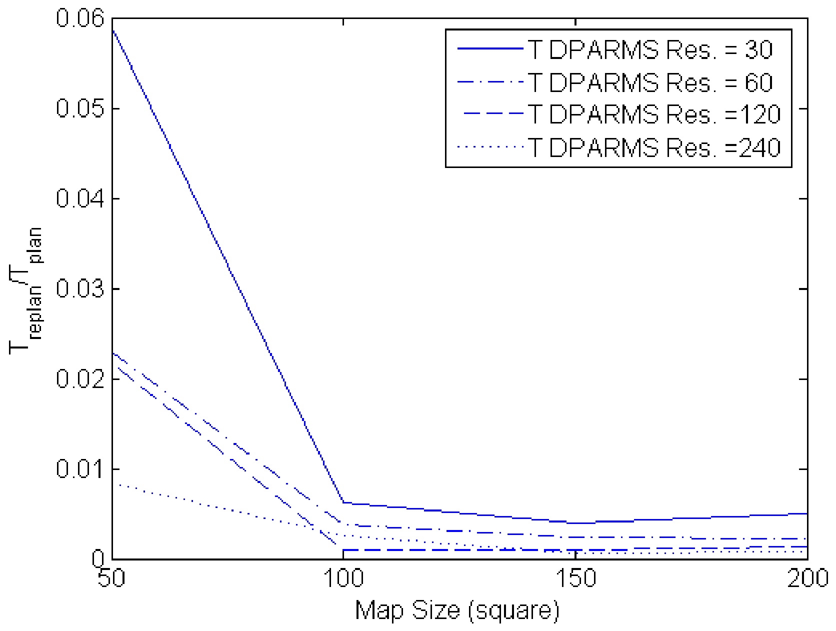


Figure 3-5: Ratio of Re-Plan Time to Initial Plan Time for Approach 2 (3-D BESTPCOST). Re-planning shows one to two orders of magnitude improvement on speed over initial planning.

3.4.7 Scaling With Solution Approach

Tests compared the time demands for the BESTDPARMS (4-D) mode and BESTPCOST (3-D) modes. Experiments demonstrate that varying the number of state parameter dimensions strongly affects the time demands of the algorithm. For these experiments, a set of 50 cost maps was generated for each combination of map size (50, 100 and 150 cells per IPARMS dimension) and DPARMS time resolution (60, 120 and 240 time units). First, the experiments ran Approach 2 on 50 maps to record the planning performance for 3-D search, as well as the durations of the resulting plan solutions for each cost map. The experiments then ran Approach 1 on the same 50 cost maps, using the Approach 2 solution plan duration as an upper bound on plan cost[1].

Figure 3-6 compares the time demands of Approach 1 (4-D BESTDPARMS) and Approach 2 (3-D BESTPCOST) for these experiments. The dotted lines correspond to Approach 1, and solid lines, to Approach 2. We immediately observe that the lower dimensionality of Approach 2 yields a sizeable time performance gain - for a given resolution and map size, Approach 2 is several times faster than Approach 1. This conforms to the notion that adding a dimension exponentially increases the number of states that a search must expand and represent in the search graph.

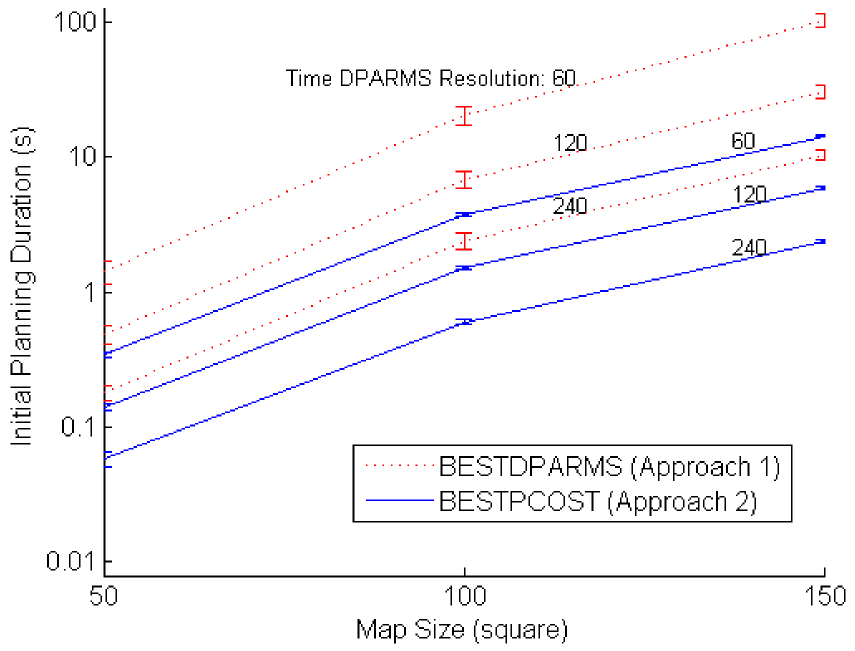


Figure 3-6: Comparison of Approach 1 and Approach 2 Time Performance

3.4.8 Qualitative Comparison of Approaches

Figure 3-7 and Figure 3-8 illustrate the qualitative differences between Approach 1 (BESTDPARMS) and Approach 2 (BESTPCOST) through plots of path, energy and progress distance. The path plots show the physical route in X-Y space taken by the plan, starting at the start position in the upper right corner, to the goal in the lower left corner. In line with the description of ISE resource variables in Section 3.1.5, each energy plot shows the profile of *minimum required energy* in the battery in order to reach the goal, as a function of plan step. *Progress distance* is a measure of the direct progress made to toward the goal. It is calculated as the Euclidean distance between the start and goal,

1. Admittedly, this may have disadvantaged Approach 1 in terms of minimum energy solution, but provided a way to more evenly compare the two approaches. Section 3.4.8 suggests that Approach 1 is able to find shorter duration solutions than Approach 2 (making this a viable test strategy for performance tests), but that the shorter duration solutions are typically far inferior in terms of energy.

minus the Euclidean distance remaining to the goal from the current robot position (for a more detailed definition, consult Appendix 2). Under this definition, progress is only made when the radius between the robot and the goal is reduced. The progress plots show progress distance as a function of plan step.

Figure 3-7 compares the BESTPCOST mode (the left column of plots - Figure 3-7a, b, c) to the BESTDPARMS mode (the right column of plots - Figure 3-7d, e, f). In this experiment, the upper bound on plan duration for the BESTDPARMS approach was set to the duration of the BESTPCOST solution (5150 time units), as in the performance comparison of Section 3.4.7. This is the BESTDPARMS Case A.

Figure 3-7a and d show the paths from both cases. Interestingly, the BESTDPARMS solution is shorter in duration than the BESTPCOST solution (4864 units) and yet wanders more by taking more steps (272 steps versus 158). The BESTPCOST solution, because it is penalized at every step for plan length and the $better(X, Y)$ function prioritizes shorter duration, deviates little from the direct diagonal path between the start and goal (8 additional steps above the minimum 150). Meanwhile, the BESTDPARMS solution is not penalized for extra steps, as long as the duration falls below the maximum. The path loiters aimlessly for some duration at the upper right of the plot and deviates from the direct diagonal for the remainder, all in order to seek low-cost energy states.

The progress distance plots in Figure 3-7c and f illustrate the tendency of the BESTDPARMS plan to be more indirect than the BESTPCOST counterpart. In Figure 3-7c, note that the progress trajectory heads at a steady slope towards the goal. In contrast, the trajectory of the BESTDPARMS solution (Figure 3-7f) oscillates at the beginning (corresponding to the period of loiter near the start) and then makes a slow but more direct approach to the goal for the remainder of the steps.

Despite being penalized for plan length, the BESTPCOST plan does a better job at minimizing energy requirements over the path. Figure 3-7b and e show the energy profiles for both solutions. In the plots, the curves rise in anticipation of a future energy cost. The curves rest at zero when energy costs are negative (net battery charging), and descend when costs are positive (net battery discharge). The BESTPCOST solution results in lower minimum required battery charge averaged over time (16.3 units versus 25.9 units) and a smaller peak energy requirement (91 versus 99 units). Presumably, this is because energy is treated directly in the objective function of the BESTPCOST approach, and indirectly through state dominance and resolution pruning in the BESTDPARMS approach.

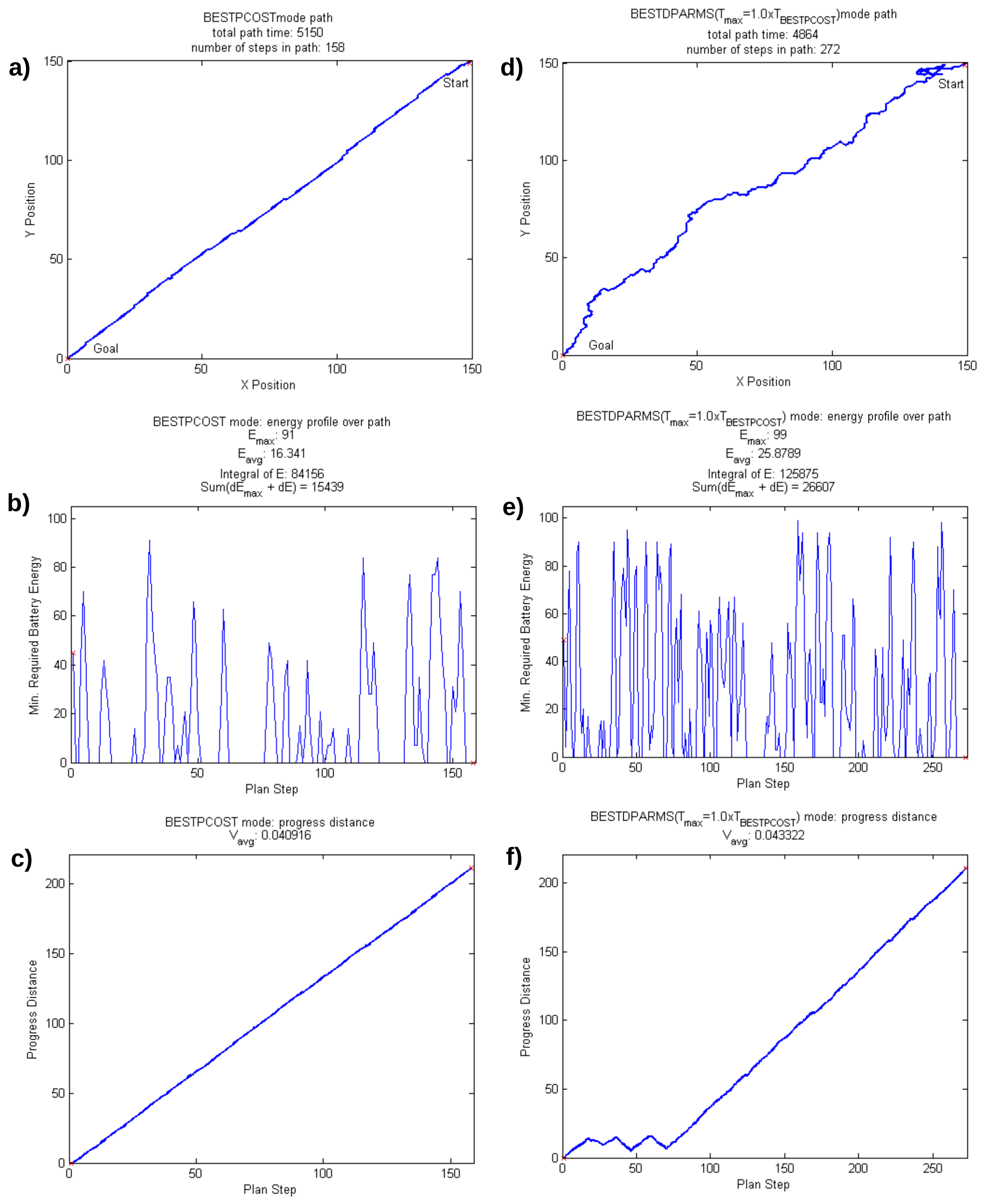


Figure 3-7: Qualitative Comparison of Planned Routes and Energy Profiles between BESTPCOST Mode (a and b) with BESTDPARMS Mode (c and d) with T_{max}=$T_{BESTPCOST}$

Figure 3-8 continues the study, with two more examples of BESTDPARMS solutions. In these tests, the BESTD-PARMS solutions were given more freedom by raising the upper bound on plan duration to 2 times the BESTPCOST duration (BESTDPARMS Case B, shown in Figure 3-8a, b, c) and 3 times the BESTPCOST duration (BESTDPARMS Case C, in Figure 3-8d, e, f). Observe first that the Case B path in Figure 3-8a loiters over a wider area than the solution in Case A in Figure 3-7d. The progress plot for Case B in Figure 3-8c confirms this - it oscillates more than the solution from Case A. This behavior results from the greater upper bound in plan duration - the plan is 440 steps long instead of 272. Interestingly, the extra duration allowed for the Case C path in Figure 3-8d does not appear to cause path further path growth from the Case B path in Figure 3-8a. The progress plots indicate differently. The progress plot for Case C (Figure 3-8f) shows an even greater period of oscillation over Case B (Figure 3-8c), even though the path didn't change from the shorter plan. The explanation is that the plan traverses the same ground more than once before moving to the goal. The planner has evidently determined the optimal loiter pattern.

The energy plots further substantiate this claim. Observe that the energy profile for Case B (Figure 3-8b) has a period of relatively low values (shown with a bracket labeled “1”) corresponding to the early part of the oscillatory behavior in the progress plot below it. This interval is followed by a sequence that completes the oscillatory steps and then continues to the goal (shown with a bracket labeled “2”). Looking closely at the Case C plot, one observes that these identical profiles also appear there. Profile 1 repeats four times in the Case C solution, and the last of these is followed by Profile 2 which continues to the goal[1]. Apparently, ISE determines the optimal path to the goal (by following Profile 2), and the optimal loiter pattern (Profile 1), which it repeats as many times as possible prior to moving to the goal. The longer the plans are allowed to be, the longer the plans linger near the start state in the optimal loiter pattern. This strategy yields improvements in time-averaged energy requirements for longer plans - the Case A, B, and C averages are 25.9, 22.1 and 18.4 units respectively. Note that none of the BESTDPARMS solutions can match the energy performance of the BESTPCOST solution.

In conclusion, it is important to state that these results are particular to the specific parameters used for the tests. Tests done on other cost maps and under different DPARMS time resolutions resulted in different behaviors. However, for this domain, tests do seem to display several patterns. The BESTPCOST approach yields more direct paths and better time-averaged energy solutions. Interestingly, the direct paths do not typically correspond to the minimum-time path. To meet the time-averaged energy performance of the BESTPCOST solution, a BESTDPARMS solution seems to require far greater time to repeatedly follow an optimal loiter pattern that drives the average energy lower.

1. A portion of Profile 2 also appears at the end of the energy plot for BESTDPARMS Case A, in Figure 3-7b.

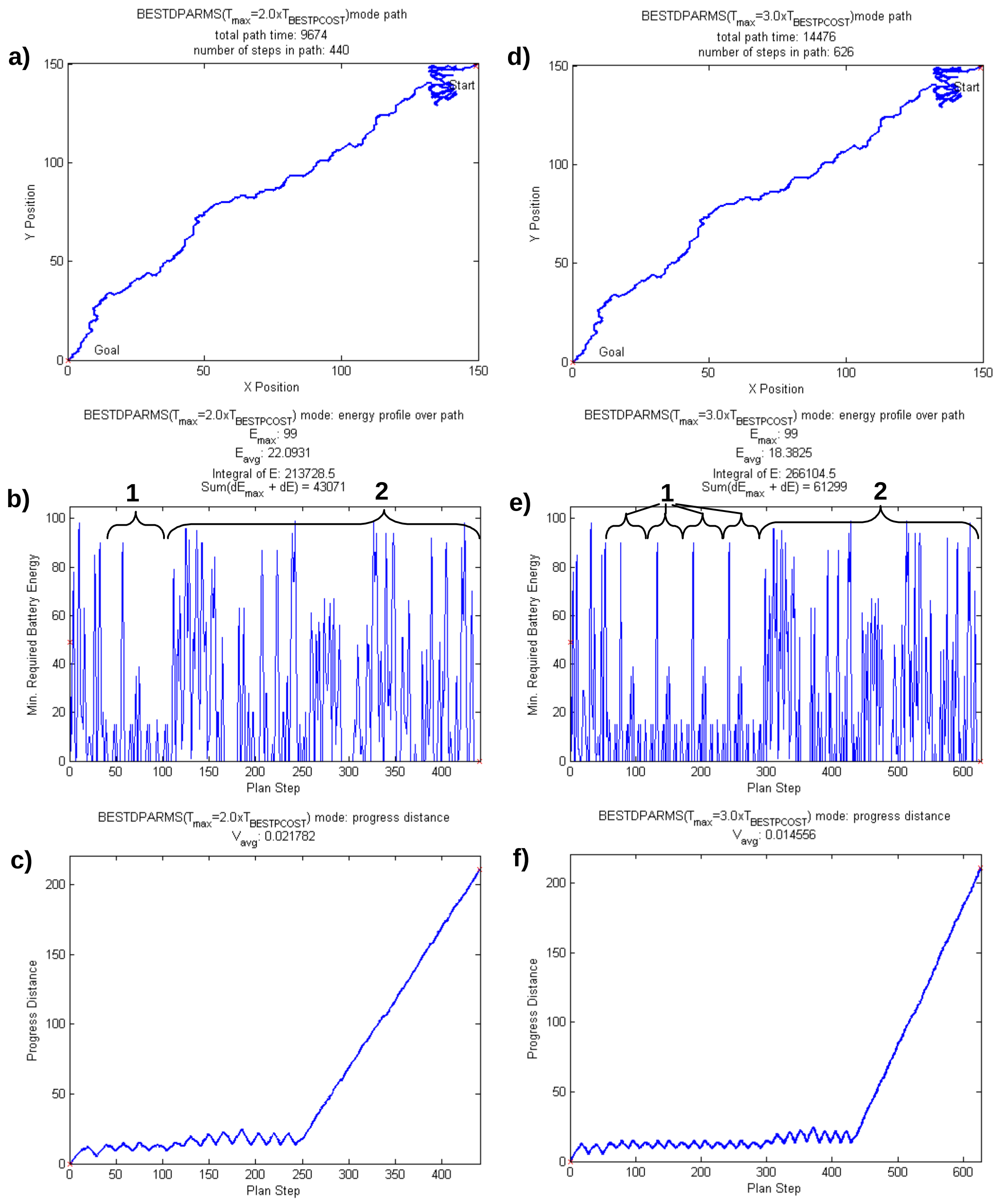


Figure 3-8: Qualitative Comparison of Planned Route, Energy Profile and Progress Distance between BESTDPARMS Mode with $T_{max}=2T_{BESTPCOST}$ (a through c) and $T_{max}=3T_{BESTPCOST}$ (d through f)

Many questions about ISE behavior remain. For instance, what is the effect of altering the $better(X, Y)$ or $dominates(X, Y)$ functions? How might they be altered in the BESTDPARMS approach to yield better results? What if time and energy costs are selected differently, for example with larger contiguous regions of low or high cost? Are there other objective functions that better achieve desired results? These questions are reserved for future work.

4. Mission-Directed Path Planning

This chapter describes the challenges posed by mission-directed path planning and introduces the TEMPEST planner, a central development of the thesis, that meets many of these challenges. The chapter defines several algorithms of increasing sophistication that enable planning in a domain combining spatial, temporal and resource dimensions, and shows how the advantageous qualities of ISE are leveraged to achieve efficient performance. Finally, the chapter demonstrates TEMPEST in a simulated multi-goal mission. TEMPEST enacts contingency planning to survive overnight in response to an unanticipated but necessary detour.

4.1 Problem Definition

The principal goal of path planning is to determine a feasible or optimal route between one position and another. What qualifies as feasible or optimal varies with the specific application problem. Mission-directed path planning seeks to derive routes for rovers exploring planetary surfaces. In contrast to other planners developed for planetary surface motion, mission-directed path planning is intended to plan over comparatively larger scales and over longer durations. Furthermore, where a majority of planetary-oriented work focuses on obstacle avoidance, the mission-directed planning domain seeks a much stronger connection with other important factors in mission planning - time, resources, operational constraints and mission returns. The following sections examine the central issues in this new domain.

4.1.1 Terrain Interaction and Obstacle Avoidance

Planetary navigation requires a robot to travel over terrain while avoiding features that impede progress to a goal position. Terrain features that might pose difficulties for navigation span a continuum of spatial dimension, from rocks at or below the scale of a vehicle to mountains many orders of magnitude larger. Much of the prior work has addressed obstacle detection, classification and avoidance for terrain features near the scale of a vehicle, called local navigation. In contrast, this work concentrates on the other end of the scale spectrum - global navigation.

In the local planning problem, kinematics and dynamics are central in determining whether terrain is trafficable. In wheeled vehicle configurations, the wheel diameter often strongly influences the size of obstacles the vehicle can drive over. A vehicle's steering scheme limits the minimum radius of curvature it can follow while driving. In some cases, dynamics are also important - a vehicle might use its forward momentum to aid in surmounting a steep slope and a fast-moving vehicle must consider its speed, center of mass and traction with the ground in evaluating the safety of candidate steering arcs. Since local navigation considers the area near the vehicle, it is often practical to represent the terrain in at a resolution smaller than the vehicle footprint. At this high resolution, path planning must consider the vehicle size and the radii of turns.

For practical reasons, global navigation de-emphasizes vehicle kinematics and dynamics. Data of the resolution required by local navigation are rarely globally available, and representing large regions at small scale is computationally impractical. Absence of high resolution data means that typical kinematic and dynamic effects cannot be adequately modeled. Global terrain is often represented at a granularity at or above the size of the vehicle. Rocks and holes at the rover scale might be totally ignored in the global context, and turning arcs might be completely encompassed within a single terrain model cell. Dynamics might affect the traversal of a small slope, but would not significantly impact the ascent of a large slope. Often, global navigation represents the vehicle as a point, and analyzes global terrain data solely in terms of gross characteristics - elevation, slope, mean obstacle density, etc.

One immediate impact of ignoring kinematics and dynamics is that the state space parameters need not include joint angles, velocities or accelerations. This makes it computationally tractable to consider other state variables like time and energy in planning.

Navigation in the large scale demands a new set of considerations. Large scale terrain units include hills, valleys, mountains, canyons and craters. In the local navigation problem, it is often convenient to classify rocks as obstacles that cannot be traversed and hence must be avoided at all cost. At the global scale, while a planner might reasonably classify steep-sided canyons or mountains as intraversible, it cannot rule out navigation over smoothly varying hills and valleys. Rather than classify entire features as obstacles, the global path planner must consider how a vehicle is affected by gross terrain properties. For example, to determine whether an area is traversable, a planner might evaluate terrain slope, vehicle heading and vehicle mass properties to determine the likelihood of tipover; gross terrain roughness to determine likely vehicle speed and mechanism wear and tear; and slope, heading, roughness and soil cohesion to estimate locomotion power. Rather than prevent access to large areas on a map, the mission-directed approach seeks to provide models of access to as much of the terrain as possible[1].

1. A problem that local and global planetary navigation share is that the places most interesting to explore and the places most challenging to navigate are often the same.

Large scale terrain affects more than just vehicle locomotion. Large positive terrain features occlude sunlight and prevent direct line-of-sight visibility to the Earth or orbiting relay spacecraft. Loitering next to a large hill could significantly reduce the duration of a communications pass or inhibit solar battery re-charging, but might keep an overheating vehicle sufficiently cool. Going to high elevation might enable a rover to view lower surrounding terrain, and might improve visibility to communications assets. Gross slope of the local terrain influences the angle of antennas or solar arrays on the rover. Driving on one side of a ridge might point a solar array towards the sun, affording a vehicle extra hours of driving time. It is the intent of this work to capture these added effects in the navigation planning process.

4.1.2 Temporal Planning

The planetary exploration domain is dynamic. Planetary motion affects sunlight, shadows, solar flux and visibility to communications assets like orbiting spacecraft or Earth-based ground stations. Local environmental parameters change as a vehicle moves from place to place. A vehicle expends and collects resources at rates that vary as a functions of the activity and the environment.

To date, path planners intended for planetary surface exploration have ignored time. Local planetary navigation planners have had two principal objectives - first, to avoid obstacles that might harm the vehicle; and second, to follow a minimum distance (or minimum time) path to a goal position. Neither objective forces a consideration of time - rock obstacles are static, and the minimum-distance path to a goal is determined entirely by the placement of obstacles in the local environment. For short-distance traverses, where local terrain parameters remain roughly constant, activity planning and scheduling software can adequately incorporate time-varying effects into predictions of daylight, shadowing and resource profiles.

In contrast, route selection in a mission-directed context must consider time. Mission success depends on having adequate resources, whose availability varies with time and position (i.e. route). Mission activities must often satisfy time-varying geometric constraints or time-based operational constraints - traverses in advance of these activities must not prevent them from being met. Depending on terrain conditions, two separate paths of equal distance might require vastly different amounts of time to follow - traversing flat, smooth terrain with few obstacles might take far less time than traveling on terrain covered with obstacles that must be either avoided or carefully crossed. In such cases, one cannot ignore time and adequately address path planning.

Planetary motion defines the gross schedule for daylight, solar flux and opportunities for communications downlinks to Earth. Exposure to the Earth or the sun is governed by whether line-of-sight visibility exists between the surface position and the source object. Lighting and communications shadows cast by large terrain features (mentioned in Section 4.1.1) vary with time. At a fixed position, shadow and sunlight schedules vary little day to day, and can be captured in a simple model that repeats. Since shadowing is a localized phenomenon, the schedule of lighting and shadow will not be repeatable for a rover that moves sufficiently far. Line-of-sight visibility to an orbiting communications relay is further complicated by the spacecraft's own motion. Consequently, the ephemerides for orbiting communications spacecraft may change dramatically opportunity to opportunity, even for a stationary vehicle. Therefore, planning for communications visibility must capture position and time dependencies. Predicting the ephemerides of the sun, Earth and orbiting relay spacecraft, and to determine whether they are shadowed or visible at specific times is critical in selecting routes that provide sufficient energy or enable communications at particular points in the path.

A mission-directed path planner must also ensure that paths obey navigational and other operational constraints. Many of these are time-dependent. A mission might dictate geometric constraints on activities, where the geometry is time-varying. As an example, consider stereo vision for autonomous navigation. Stereo vision is strongly affected by lighting levels. Glinting sunlight on camera lenses and entry of the solar disk into the camera field of view causes "phantom obstacles" to be generated in depth maps, or causes the rover to be blinded. To prevent glinting and blinding, locomotion actions might be disallowed when the sun is within a specified angle from the camera boresight. Assuming the cameras point at or below horizontal, driving would be prevented in certain directions in the morning and evening, but would be unconstrained during midday. Restricting photographic measurements to times when the sun is shining on the desired target is a second example of a constraint on time-varying geometry. Operational constraints may also be purely temporal. A communications downlink to Earth might be geometrically feasible over a range of times, but operationally feasible only over a shorter time window allocated to the mission. Low power, low activity phases like hibernation might only be allowed during nighttime.

A central theme of this work is the modeling and long-distance path planning in the presence of time-dependent effects, time-varying geometry, and variability of traverse speed as a function of terrain or other effects.

4.1.3 Resource Planning

A central problem of mission planning is ensuring the availability of resources for activities. A resource is a quantity that must be expended to achieve goals, but that is in limited supply. There are different types of resources. *Metric resources* are storable quantities, for example time, energy or fuel, that are expended through activities that require the resource. From a planning perspective, metric resources can also be more abstract - the finite lifetime of a motor, or the temperature margin on a thermally-sensitive instrument. Metric resources further subdivide into *monotonic resources* and *non-monotonic resources*. Monotonic resources can only be expended; non-monotonic resources can also be replenished through collection activities. Metric resources are described by continuous, real-valued variables. *Unit resources* describe components that are fully committed during an activity and become uncommitted at the termination of the activity. These are described by Boolean variables describing either commitment or availability, as in a camera or a motor. This research addresses metric resources and not unit resources.

Metric resources are a critical factor in planning paths in the mission context. Without the required energy, a vehicle cannot perform its primary activities. The route a vehicle follows from place to place has a substantial effect on both the expense and collection of resources. Contrary to the assumption of many simple path planners, the shortest route is not necessarily the least costly. Steep, soft, slippery or rocky terrain requires more power than level, firm and flat terrain. For vehicles that collect resources like solar energy from the environment, the choice of path can dramatically influence the available resource levels. All other parameters being equal, a path in shadow cannot yield the same energy as a path that is fully exposed to the sun.

A mission-directed planner must, at minimum, guarantee satisfactory resources for desired mission activities, and may also provide a plan that is optimal in terms of resources. It must consider the balance between resource consumption and collection, and, based upon specified activity models and stated requirements, provide the necessary resources at all phases of the plan. Another focus of this work is to enable metric resource planning to achieve satisfactory or optimal energy profiles in the context of planetary mission exploration.

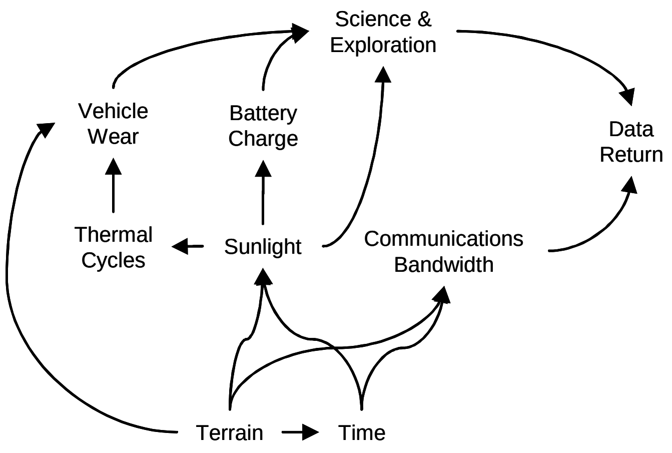


Figure 4-1: Coupling Between Terrain, Time, Resources and Mission Return

4.1.4 Coupling of Variables

The effects of terrain, time, resources and mission return are highly interwoven. None can be considered in isolation from the others. The diagram in Figure 4-1 illustrates some of the common interrelationships in surface mission operations. Terrain affects the speed of travel and causes vehicle wear. It also causes shadowing of sunlight and communications. Sunlight provides solar energy for battery charging, and enables imagery for navigation and science, but also causes thermal cycling and resulting wear on the vehicle. The mission objectives, science and exploration activities, are supported by battery charge and constrained by vehicle wear. Once data has been collected, communications with Earth enables its transfer to an operations team. Therefore, to achieve the ultimate objective of retrieving data from the robot, mission planning must consider a variety of factors in concert. The challenge for automated planning research is to provide a method for solving this problem in a computationally practical way.

4.2 TEMPEST

TEMPEST (TEmporal Mission Planner for the Exploration of Shadowed Terrain) is a path planner designed for mission-directed reasoning. It combines five models that define the relevant features of the mission-directed planning domain, and uses the Incremental Search Engine (ISE) to search for plans that achieve mission objectives while satisfying operational constraints (see Figure 4-2). The following sections describe the TEMPEST models, and explain how they collectively contribute to the more formally-defined functions required by ISE.

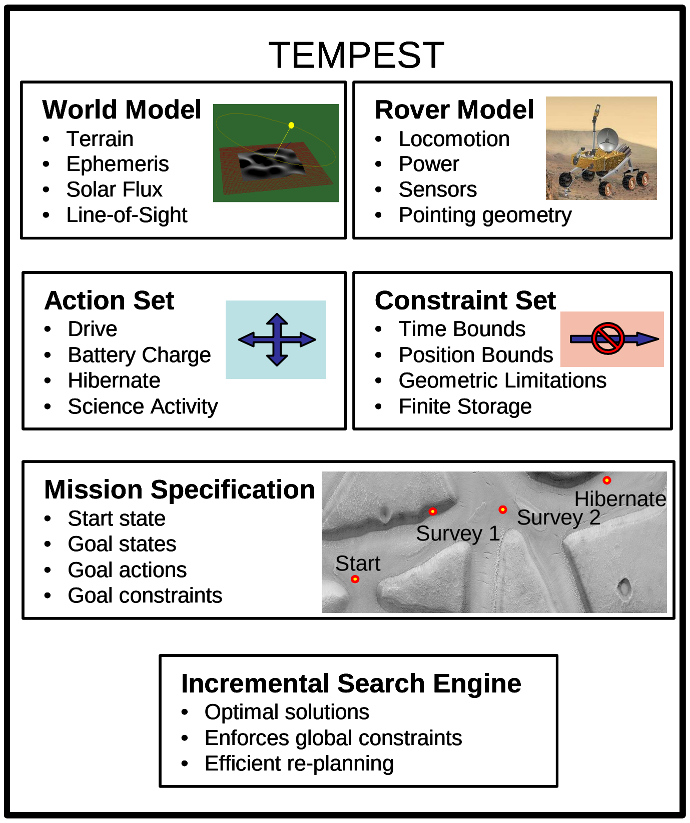


Figure 4-2: Principal TEMPEST Components

Mission-directed path planning must consider the interactions between several elements:

- Planetary environment through which a path must be planned
- Robotic vehicle that is operating

- Actions the vehicle can take that are relevant to the problem
- Physical and operational constraints on the available actions
- Mission objectives

TEMPEST composes five models corresponding the above elements. A *World Model* captures relevant environmental phenomena for the planetary surface; a *Rover Model* describes relevant components of the vehicle operating in the planetary environment; an *Action Set* comprises the activities the vehicle can execute to traverse across terrain, maintain resources, and achieve mission objectives, and the effect they have on vehicle state; a *Constraint Set* imposes restrictions on the available actions, in terms of the state of the *World Model* and *Rover Model*; and finally, a *Mission Specification* describes the initial state of the vehicle, the immediate goals of the mission, and the specific actions and conditions under which the goals can be satisfied.

The models must be tailored to the planning problem to encompass all of the desired rover-environment-mission interactions. However, as stated earlier, it is not computationally practical to represent the domain at high resolution when plans are intended to span kilometers and tens of hours. Hence, the models are purposely coarse. They provide reasonable projections of action outcomes under various environmental conditions, but at a resolution that permits sufficiently high performance on a rover processor. Each of these models is described in greater detail in Sections 4.2.1 through 4.2.5.

The models are the foundation for defining the ISE state space, the start and goals, transition arcs between states, and the constraints on them. Section 4.2.6 briefly describes how model information is used to define these ISE domain components.

4.2.1 World Model

A model of the planetary environment is fundamental in producing plans that avoid hazardous terrain, consider sunlight exposure and follow the most time and energy efficient routes. The *World Model* captures the relevant features of the planet environment in which a vehicle operates. It is one of two basis models for defining the state space, for computing arc transition costs and for determining the conditions under which constraints are satisfied or violated.

To reflect local conditions defined by the underlying models, the *World Model* is set to a particular state, defined by a position on a planetary surface, time and surface orientation. With each update to the state, the *World Model* calls on its underlying components to compute the local conditions. It currently includes, but is not limited to, the following components:

Geodetic Reference - All maps in the *World Model* are referenced to a geodetic reference, a reference biaxial ellipsoid (ellipsoid of rotation) that approximates the shape of the planetary body, from which transformations between coordinate frames can be defined [7]. The coordinate frames that apply to all maps are:

- Planet Cartesian: a frame whose origin is at the center of the reference ellipsoid, and whose axes are defined by the following:

  - X: in the plane of the equator of the reference ellipsoid, and in the plane of the prime meridian of the system.
  - Z: the axis of symmetry of the ellipsoid, and parallel to the axis of rotation of the planetary body.
  - $Y = Z \times X$ to make a right-handed frame.
- Geodetic: a system comprising latitude, longitude and altitude. The geodetic system has two variants, based loosely on the NASA Planetary Data System convention [27], planetocentric and planetographic. In the *World Model*, they are defined as follows:

  Planetocentric
  - Latitude:
  - Longitude:
  - Altitude:

  Planetographic
  - Latitude:
  - Longitude:
  - Altitude:
- Topocentric: a Cartesian frame whose origin is a point on the planetary surface. Its axes are defined as follows:

  - X: in the plane tangent to the surface of the reference ellipsoid, in the direction of constant latitude East.
  - Z: the reference ellipsoid surface normal.
  - $Y = Z \times X$ such that Y points North along a meridian.

Elevation Map - A map of the operations area encodes elevations, above the reference ellipsoid, in a two-dimensional grid of positions. Spatial resolutions for this data are typically 10-30 meters per pixel, far larger than the vehicle footprint. Though the model allows smaller ratios of map pixel to vehicle size, a high ratio preserves the assumption that vehicle turning radius is insignificant with respect to cell size. The elevation map also computes vectors between points on the map.

Slope Map - The *World Model* derives slope and slope aspect from the elevation map, and encodes each quantity at the same spatial resolution as in the elevation map. In addition, the slope map computes the transformations between the topocentric frame and two other frames:

- Gradient: a frame whose origin coincides with the origin of the topocentric frame, and whose axes are defined by:

  - X: in the plane of the local terrain surface, in the direction of steepest ascent (gradient).
  - Z: the terrain surface normal.
  - $Y = Z \times X$ to make a right-handed frame.

- Pointing: a frame that describes the orientation of an object on the terrain surface. Its origin coincides with that of the gradient and topocentric frames, but its axes are defined by:

  - X : in the plane of the local terrain surface, at an arbitrary angle from the direction of steepest ascent.
  - Z : the terrain surface normal.
  - $Y = Z \times X$ to make a right-handed frame.

Ephemeris - The ephemeris model predicts the vector of source bodies in the planet Cartesian frame of an observing body at a particular time. The ephemeris model uses CSPICE, ephemeris generation software that provides relative position and orientation for all major bodies in the Solar System [1]. In counterpoint to the guiding principle of coarse modeling, CSPICE is a very accurate tool, accounting for speed-of-light delays and stellar aberration in determining a body's apparent location. The ephemeris time standard - barycentric dynamical time - is the basis time reference in the *World Model*.

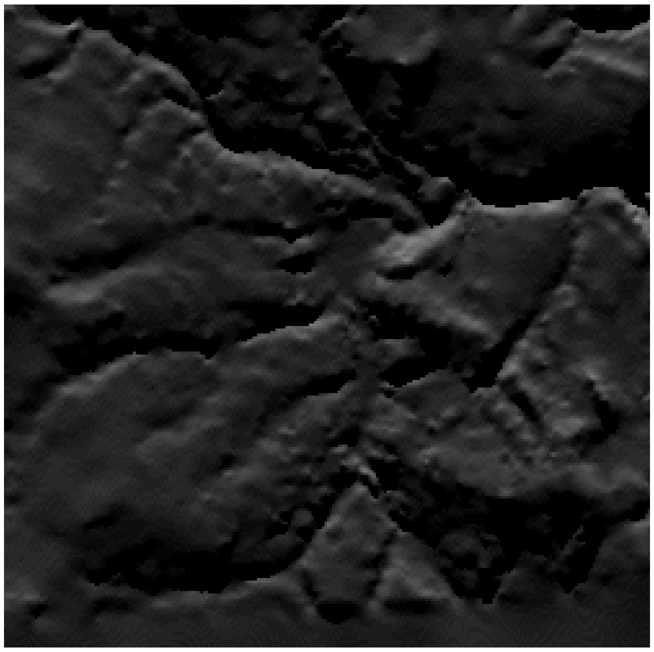


Figure 4-3: Example LOS map for sunlight on natural terrain, in this case a system of canyons exposed to the sun from the direction of the top of the image.

Line-of-Sight Maps - Line-of-sight (LOS) maps encode the elevation angle of a source object above the local ground plane as defined by the slope map (see Figure 4-3 for an example). They also map shadowed locations, where the source is below the ground plane or occluded by other terrain features. LOS maps currently represent incident sunlight for the purpose of modeling solar energy, lighting and shadowing. However, they could also represent line-of-sight to orbiting spacecraft or visibility to fixed points elsewhere on the terrain surface. A ray tracing algorithm projects from the source position (e.g. a Solar System body) onto the terrain model. Sequences of LOS maps, at regular time intervals, represent time-varying visibility.

Solar Flux - The incident energy per unit area is modeled by the peak flux (as experienced under perpendicular incidence) multiplied by the sine of the sun elevation angle to capture foreshortening effects. Atmospheric attenuation must be captured in the peak flux value - the *World Model* currently assumes no variation in flux with angle from zenith. We currently ignore all other effects. This model is used principally for the computation of available solar power, but might also be used to compute the sun's thermal influence on vehicle components.

4.2.2 Rover Model

The path planner must be able to predict the effects of activities undertaken under different environmental conditions. The basic unit of the *Rover Model* is the rover component model. Components model mission-relevant units or capabilities of the rover, and include parameters relevant to operational constraints on the rover. Each component in the *Rover Model* can be activated or de-activated based on the activity being performed. Active components can be set to a continuous range of activity level, defined by a duty cycle. The *Rover Model* aggregates components and utilizes the *World Model* to predict how activated components affect the rover internal and external state.

As with the *World Model*, the *Rover Model* must be set to a particular state which in turn defines the behavior of rover components. In this thesis, since the emphasis of TEMPEST has been on path planning that enables energy management, the *Rover Model* uses only battery energy state. The *Rover Model* predicts power load as a summation of the powers from each activated component, scaled by their duty cycles. However, one could envision modeling other resources similarly, for example component wear, component thermal state, or available computer memory. Science data might be analogous to energy, and onboard memory the equivalent of a battery - science data collection and communications with Earth would then be the activity equivalents of battery charging and discharging, respectively. The following are a few examples of rover components that are possible in the *Rover Model*:

Locomotor - A component that models the speed and power of a vehicle as a function of terrain, and determines whether or not terrain is traversable. One simplified instantiation of the model is parameterized on vehicle mass, drivetrain efficiency, and effective coefficient of friction. Other, more sophisticated models might incorporate models of bulldozing resistance, rolling resistance and slippage.

Battery - A component that models energy storage capacity. A basic model might simply encode minimum and maximum bounds on charge. More sophisticated models could incorporate charge and discharge rate limits, transmission losses, and might model individual cells in the battery.

Solar Array - The solar array is a pointed component model. A simple model encodes the array normal vector with respect to the vehicle frame, the array area, and the solar cell efficiency. More complicated models might incorporate models of individual cells, the strings of cells, and degradation effects due to dust collection or radiation.

Power Load Component - A generic component type, the power load component describes a fixed power load (positive) or power source (negative) in the system. The component models steady loads for onboard electronics (positive loads) or the power coming from a radioisotope thermoelectric generator (RTG - a negative load).

Field Of View Component - A generic pointed component type that models the field of view of sensing devices on the vehicle. Used in defining geometric constraints, this component might define cameras, sun sensors or communications antennas, and is not required to specify the component power.

To date, the emphasis of TEMPEST planning has been on path planning that enables energy management. Therefore, most rover components predict power load as a summation of the powers from each active component, scaled by their duty cycles. One might envision modeling other resources similarly. Components might also model thermal energy or component wear as a function of *World Model* conditions.

4.2.3 Constraint Set

Constraints in TEMPEST encode the set of world and rover state values under which an action is illegal. They can represent either physically impossible conditions, or conditions that are undesirable operationally. The *Constraint Set* aggregates a number of individual constraints, each of which can be activated or de-activated for selective application to different actions.

To test for constraint satisfaction or violation under certain conditions, the *Constraint Set* sets the *World Model* and *Rover Model* states to the test condition, then checks each activated constraint for a violation. The individual constraints access the *World Model* and *Rover Model* local state parameter values to compute Boolean constraint violation functions. Because they depend only on current world and rover state, *Constraint Set* constraints are *local* as opposed to *global*. The following examples give a flavor for the kinds of constraints that are possible:

Position - Actions might either be restricted *to* or restricted *from* operating within a particular area on a map. The position constraint enables a user to specify a set of positions, and whether the positions define a legal or illegal region. This constraint is useful for defining hazardous regions not otherwise modeled in the *World Model*, or perhaps scientifically interesting regions within which a particular set of measurements is useful.

Maximum Slope - Slopes present a hazard to rover driving. Driving on a steep slope can risk vehicle tipover. Even if tipover is not an issue, steep slopes may be impossible to climb due to limited traction. In the simplest model, a user can select a maximum slope that is legal to operate on. This model calls exclusively on the *World Model*.

Direct Line of Sight - Line-of-sight (LOS) geometry is important for a number of applications, including solar power collection, communications and remote sensing. A source is within LOS of the vehicle if a ray from the source to the vehicle does not intersect the terrain surface. The LOS constraint allows a user to specify a source object in the *World Model* with which LOS geometry must be evaluated during search. The constraint can be defined either to succeed or fail if LOS conditions exist. A user might define a “shadowed” constraint to define the illegal conditions for solar charging, or a “sunlit” constraint to define illegal conditions for a thermal cooling action. Note that a “shadowed” constraint does not distinguish between simple terrain occlusion and below-horizon conditions. Similar constraints could be generated for other sources (e.g. “out of view of Earth”, “in view of Mars Odyssey”, or “out of view of alluvial fan X”).

Elevation Angle - The elevation angle constraint limits the geometry of a Solar System object with respect to the local horizontal plane. A user must define the source object, the threshold elevation angle and whether the illegal range is above or below the threshold elevation. The primary use of the elevation angle constraint is to define the elevation angle threshold for “daytime” or “nighttime” conditions.

Field of View - Sensor geometry can often be defined in terms of a boresight vector and its field of view (FOV), an angle about the boresight defining its cone of sensitivity. The user defines an FOV constraint by selecting an FOV component from the *Rover Model* and a source object from the *World Model* (for example the sun), and designates whether the constraint is violated when the object falls within or outside the FOV. One might define the range of sun-boresight angles for which a sun sensor returns accurate vehicle heading estimates, or the spacecraft-boresight angle range when communication are possible to an orbiting relay satellite.

Time Bounds - A mission might require or disallow actions to fall within a fixed time range. This constraint permits actions to be constrained within or outside a time bounds.

Battery Charge - This constraint specifies the legal range of battery state-of-charge. To specify this constraint, a user associates the constraint with a specific battery model defined in the *Rover Model*. Hence the range is defined by the *Rover Model*. A particularly power-hungry activity might require a high state of charge. Alternatively, stopping to charge a battery using solar energy might only be justified when the state of charge is below a certain threshold.

Multiple constraints can be applied to a single action to dictate more complicated constraint conditions. For example, one could define a "Prevent Sun Blinding" constraint, to specify the conditions under which a camera is blinded by the sun. It might combine the "Direct Line of Sight" and camera "FOV" constraints, since the FOV constraint by itself would ignore whether the sun is actually visible or occluded from view. A constraint to identify times near sunrise and sunset might combine two sun elevation constraints, for example "Sun Above the Horizon" and "Sun Below 10 Degrees Elevation".

4.2.4 Action Set

The *Action Set* aggregates the actions that are most relevant to the planning problem. The central theme of path planning is to plan for motion through an environment - therefore motion actions are required in all domains. Other action types are optional. In domains where resource management is important, the *Action Set* might include actions for resource collection (e.g. battery charging) and must represent salient resource consumption activities. If mission goal-related actions (e.g. science data collection) consume significant time or resources, or if constraints imposed on them might affect the route or schedule, then they must also be included.

Table 4-1 lists the data required to define a TEMPEST action. The *World Model* and *Rover Model* play a substantial role in the behavior of the action. Given the *target* change in state (Line 1), the active rover components (Line 2) determine the change in other state variables. For instance, if a Drive action targets a change in X and Y position, the rover components determine, through speed and power equations, the resulting change in time and battery energy. Through Line 4, constraints can modify the behavior of the *Rover Model* by allowing actions in some states and preventing them in others. Line 3 is a condition that allows TEMPEST to remove vehicle heading from the search space. In most cases the optimal rover orientation is either a direct function of the target change in state, or more a function of local state than of global path state history.

Table 4-1: Data Required to Define a TEMPEST Action

1. A target change in state. Mobile actions specify a target change in position. Stationary actions result in zero net change in position, and so specify changes in other state parameters.

2. A list of active *Rover Model* components and their duty cycles. The components must enable the target state change, and must uniquely determine the resulting change in all other state parameters.

3. A function $\theta = f(s, a)$ that uniquely determines the vehicle orientation as a function of state.

4. A list of active *Constraint Set* members to be actively enforced.

An important limitation is that TEMPEST only solves for plans that are sequences of fully-ordered actions. This research has not actively sought a scalable means of representing parallel actions. Currently, parallel actions can be approximated by combining the behaviors of each action into a single action model, but representing all possible action combinations would result in an exponential growth of the action space.

The following examples of TEMPEST actions illustrate how the conditions in Table 4-1are met:

Drive - An action to enable vehicle mobility in a plan. The data required to define a drive action are listed in Table 4-2. The simplest drive actions enable motion to each of the eight neighbors in an eight-connected grid map. Each cell destination defines a separate action. The vehicle orientation is a direct consequence of the position target - the vehicle must face in the driving direction to achieve the target cell[1]. Vehicle speed is defined by local terrain parameters and rover parameters.

Table 4-2: Drive Action Data

Data Type	Data
State Change	Change in position $\Delta x, \Delta y$
Minimum Active Rover Components	Locomotor that specifies vehicle speed $v = f(s, a)$ .and power $p = f(s, a)$
Heading Function	$\theta = f(\Delta x, \Delta y)$ such that vehicle faces the target.
Active Constraints	Arbitrary (Maximum Slope, Position, etc.)

One or more rover components must yield the change in time and resource state variables for the target position change. Specifically, the component must specify the vehicle speed and power given the state (e.g. as a function of local terrain, available solar power and sunlight, vehicle mass, etc.). Fixing the speed is not a substantial limitation of the model. Current planetary rovers are typically designed to drive at a single speed. Control strategies may command lower speeds when following sharper turns or when in hazardous terrain. Though neither turning radii nor local obstacles are explicitly represented at the low resolution of global planning, their long-term effects can be captured in an average speed model. If a single speed does not adequately model the vehicle's motion, one can define multiple drive actions that have the same position change but utilize locomotor components with different speeds.

Constraints for drive actions are optional. However, it may be useful to constrain driving to below a maximum slope, to avoid hazardous terrain, or to times and headings that keep the sun disk out of the field of view of the navigation cameras.

1. Note that for omni-directional vehicles, the drive action must define an explicit function for vehicle orientation.

Solar Charge - An action to enable stationary solar-powered battery charging. Being a stationary action, the solar charge action must target a duration or a target energy. At minimum, the action must call on two rover components - a solar array to collect energy, and a battery to store incoming energy. The vehicle heading for a solar charge can be defined to optimize the incoming solar energy over the duration, as a function of terrain slope, absolute time, and the orientation of the solar array with respect to the rover frame.

Table 4-3: Solar Charge Action Data

Data Type	Data
State Change	Duration Δt or Energy Δe
Minimum Active Rover Components	Solar Array that specifies solar power $p = f(s, a)$ Battery to store incoming energy.
Heading Function	$\theta = f(t, slope, az, el)$ such that vehicle faces the optimal direction for solar power collection.
Active Constraints	Arbitrary (Sun LOS, etc.)

As for drive actions, constraints for solar charge actions are optional, but can be added to limit their use to certain state conditions (e.g. only when the sun is visible).

Hibernate - An action to enable stationary operations at low power levels. This action is actually a variation of the Solar Charge action, but uses different *Rover Model* components (e.g. low power electronics), has potentially a different duration target, and employs a different set of constraints (e.g. Nighttime).

Science Data Collection - An action to model the activity of science data collection within a single position grid cell. Since the action remains within a single position cell, Science Data Collection is also a variation on the Solar Charge action. It uses *Rover Model* components to approximate the power of various instruments, overhead electronics power, and perhaps power for limited locomotion about the site. The activated constraints for this action type would have to match the requirements of the particular measurements being employed. One could envision designing constraints to impose lighting conditions, certain geometric conditions, or even thermal conditions at the site.

Via Point - A placeholder action, inserted at a goal position, that has no effect on state, but satisfies the goal of reaching a particular position. The following section explains why this type of action can be useful.

4.2.5 Mission Specification Set

The *Mission Specification* encodes the basic objectives for the planning run - the starting conditions for the plan and a list of goals to be achieved.

The rover start specification defines the start position and time in *World Model* coordinates, and the current resource levels, defined by the *Rover Model*, that cannot be exceeded by the starting resource levels of any feasible path.

The goal specification is a fully-ordered goal sequence, each defining, at minimum, the position of the goal and the goal action - an action defined in the *Action Set* that must be executed at the goal position to achieve the goal. Often, a mission will specify science or exploration activities at specific locations, or resting points for overnight periods. The previous section described Science Data Collection and Hibernation actions for these times. In other cases, a goal will purely be a via point en route to some other location. In this case, Via Point action can be used. As with any action, goal actions can constrain the states over which the actions are legal, via the *Constraint Set*. Optionally, goals can independently specify allowable termination time bounds or minimum resource levels.

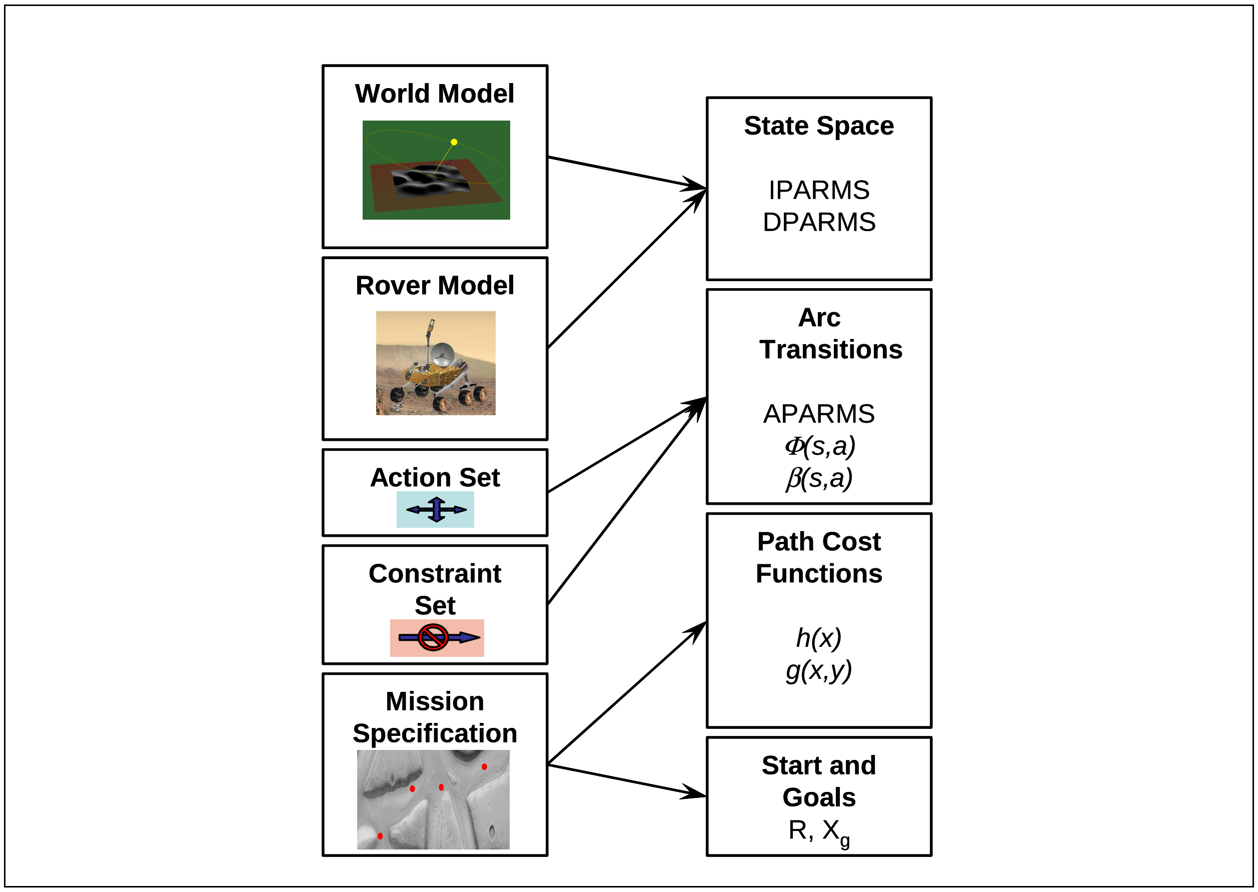


Figure 4-4: ISE Domain Definition through TEMPEST Models

4.2.6 Incremental Search Engine

The five TEMPEST models collectively define the ISE planning domain - the state space (IPARMS and DPARMS), the arc parameters (APARMS) and arc transition functions, the objective and heuristic functions, the start and goals and other domain-specific functions (see Figure 4-4). For greater detail on ISE, please refer to Chapter 3.

The ISE state space derives primarily from the states variables defined by the *World Model* and *Rover Model*. At minimum, the *World Model* contributes two position dimensions X and Y, and an absolute time dimension T. The position variables X and Y form the ISE IPARMS. The time dimension T is the only required DPARMS parameter. TEMPEST time units are seconds, but time equivalence class partitions are application-specific and typically far more coarse, for example 10-30 minutes.

The *Rover Model* contributes metric resource parameters. Resource parameters can either be DPARMS and limited under local constraints or, in circumstances where the resource value relates directly to path cost, they can be represented as global constraint parameters within the objective function (see Sections 3.1.3-3.1.5 for more detail). The resource of choice in this thesis is rover battery energy; however, TEMPEST could be re-configured to plan for other resources, like component life, margin on thermal thresholds, or onboard memory.

The TEMPEST *Action Set* and *Constraint Set* collectively define arc parameters (APARMS) and arc transition functions β and Φ. Actions in the *Action Set* define the change in state given an initial or final state. The *Constraint Set* determines the conditions under which actions are legal or illegal, enabling the arc transition functions to reject violating arcs. In general, ISE uses APARMS to store all parameters that influence arc transitions. TEMPEST uses APARMS exclusively to encode parameters that are expected to change during plan execution, and hence prompt re-planning. The static parameters are stored in a data structure universally accessible by the state transition functions.

The *Mission Specification* defines the goal states and actions and the start state for the ISE search. Recall that ISE plans paths to one or more goal states. As will be shown in later sections, TEMPEST assigns multiple, time-distributed goal states to goals that can be feasibly completed over more than one time equivalence class. During search, each time-distributed goal state is treated independently and can be expanded to yield new states. The start state specification defines the termination conditions for the search - encoded in the ISE function $feasible(X)$.

The path cost functions $h(X)$ and $g(X, Y)$ depend indirectly on the *Mission Specification*. The *Mission Specification* defines which cost is important to the planning problem. In the purest sense of mission-directed planning, as defined in the Introduction, cost is in terms of the reward earned through achieving mission goals. However, in many circumstances other costs based on resource parameters or other variables have direct mission relevance. This thesis does not address reward-based planning. That said, TEMPEST would not require significant modification to couch all goals and path costs in terms of reward.

4.3 Algorithm

This section describes how TEMPEST uses ISE to solve for plans. The basic unit of TEMPEST planning is the *path segment* (or segment for short) - the interval between a pair of goal positions for which a plan must be solved. For each segment, TEMPEST creates a separate instantiation of ISE that is dedicated to solving that sub-problem. Each ISE instance must be initialized with goal and start information. Once initialized, TEMPEST uses the ISE instances to generate paths between goals. Sections 4.3.3 and 4.3.3 describe the initialization and planning algorithms for a single goal. Sections 4.3.4 through 4.3.6 build on this base case to handle multiple goals and goals with completion time bounds.

4.3.1 Definitions

Table 4-4 lists a number of other informal definitions for symbols and functions that appear later in the algorithm listings.,

Table 4-4: Informal Definitions of Symbols and Functions Used in the TEMPEST Algorithm Description

Item	Definition
CREATE_ISE()	A function that yields an instance of ISE planner.
DELETE_ISE(I)	A function that deletes an ISE instance.
D_P	The progress distance (see Appendix 2, equations A2-1 - A2-3).
F_P	The progress fraction (see Appendix 2, equation A2-4).
G	An individual mission goal specification. In general, a goal specification is defined by the tuple $x_g, y_g, \bar{T}_g, \delta t_g, r_g, c_g$, the goal position, the legal time interval for goal completion, the goal action duration, the goal minimum resource level, and the goal “final” cost. The time interval $\bar{T}_g$ can be either unspecified ($\bar{T}_g = (-\infty, \infty)$) or partially specified ($\bar{T}_g = (-\infty, t_f)$ or $\bar{T}_g = (t_i, \infty)$).
.Γ	A fully-ordered sequence of N goals $(G_1, \ldots, G_N)$ where elements are listed in the desired order of completion, and each goal element $G_i \equiv x_i, y_i, \bar{T}_i, \delta t_i, r_i, c_i$
$init(I)$	A function that returns $TRUE$ if the ISE instance I has just been initialized, and $FALSE$ otherwise.
$last(\Gamma)$	A function that, for a sequence of goals Γ, returns the index for the last unplanned goal. If some number of goals $n < N$ at the end of the sequence still maintain valid plans, $last(\Gamma)$ returns $N - n$.
$next(\Gamma)$	A function that, for a sequence of goals Γ, returns the index for the next goal to be achieved. If some number of goals $n < N$ has already been achieved or abandoned, $next(\Gamma)$ returns $n + 1$.
$\rho(X, Y)$	A function that returns the minimum traverse distance between two states X and Y.
R	The minimum possible traverse distance to intersect all goal positions (see Appendix 2, equation A2-1).

Table 4-4: Informal Definitions of Symbols and Functions Used in the TEMPEST Algorithm Description

Item	Definition
S	The mission start specification, defined by the tuple $\langle x_s, y_s, t_s, r_s \rangle$, the position, time and resource parameter values for the start state.
SET_GOAL(G)	An ISE function that sets the augmented goal state $g = \langle x, y, t, r, c \rangle$, the goal position, time, resource level and "final" cost. The state is placed onto the OPEN list with $h(g) = c$.
$\bar{T}$	A time interval (t_i, t_f), where t_i is the first time in the interval, and t_f is the final time. The functions $open(\bar{T})$ and $close(\bar{T})$ yield the times t_i and t_f respectively.
ΔT_{res} i	The resolution of the time state parameter DPARMS equivalence class.
v_{max}	The maximum possible rover speed, used to compute the minimum mission completion time Δt_{min}, and is a function of the *Rover Model* and *World Model*.
v_{min}	The minimum allowable average rover speed. This parameter is used to compute the maximum allowable mission completion time Δt_{max}, setting an upper-bound on a plan's path length and loiter time. v_{min} is typically fixed for a given application.

4.3.2 Single-Goal Planning

In TEMPEST single-goal planning problem (Figure 4-5), the basic objective is to solve for an optimal plan that travels from a start S to a goal G, and executes a goal action before terminating. TEMPEST initializes a single segment planner in a function INIT_SEGMENT(S, G). The algorithm for INIT_SEGMENT(S, G) appears in Table 4-5.

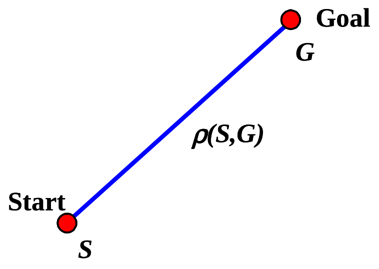


Figure 4-5: Single-Goal Planning Problem

Before describing single-goal planning, we introduce Figure 4-6, which has six frames depicting TEMPEST single-goal segment initialization and planning. Each frame is a plot of time (horizontal) versus progress fraction F_P (vertical). The origin of each plot represents the start state, with a time just before the start time t_s, and a distance of zero.

On the distance axis, the high point represents the Euclidean distance to the goal. The slope of a trace in a plot indicates the speed of progress towards the goal. The time axis may span one or more days, as denoted by the light and shaded regions corresponding to "noon" and "midnight". The following paragraphs refer to these plot diagrams to illustrate the TEMPEST algorithm.

Function INIT_SEGMENT(S, G) begins at Table 4-5, line L1 by defining the allowable plan start time interval $\bar{T}_s$, which is exactly ΔT_{res} in duration and centered on the start time. This is depicted graphically in Figure 4-6a) by the short, thick line segment surrounding the start time. Line L2 computes the minimum distance R between the start and goal. Lines L3 and L4 compute the minimum *possible* and maximum *allowable* goal completion times for the segment. The maximum allowable time accounts for the worst-case traverse duration and the goal action duration, starting from the close of the start time interval. In contrast, the minimum possible time starts at the opening of the start time interval, but neglects the goal action duration. This allows for the possibility that the goal action is abandoned during execution. Figure 4-6a) depicts the maximum speed line (more steep) and minimum speed line (less steep) that define these times - dashed lines that ascend from the limits of the start time interval.

These times are endpoints for two time intervals that are used repeatedly in TEMPEST planning - the *reachable* time bounds $\bar{T}_{reach}$ and the *allowable* time bounds $\bar{T}_{allow}$ (line L5). The reachable time bound opens at the earliest possible goal completion time and extends to infinity. It represents physically possible outcomes. The allowable time bound has an unlimited lower bound, but sets an upper limit on times considered for the search. The goal completion time interval $\bar{T}_g$ is computed as the intersection of the reachable and allowable completion times (line L6). In Figure 4-6a), the thick, horizontal line segment at the goal distance shows the range of allowable goal completion times. Note that the goal time interval extends beyond the maximum traverse time to account for the goal action duration.

It is important to note that although a minimum average rover speed is used to limit the range of allowable goal completion times, TEMPEST does not further limit the average speed of paths. Figure 4-6b) shows the feasible distance/time space as bounded by two lines - the original line of maximum rover speed, defining the earliest reachable approach; and a second, new line extending downward from the latest possible goal position arrival time (at the same speed), defining the latest allowable approach to the goal.

Table 4-5: INIT_SEGMENT(S,G) Algorithm

L1	$\bar{T}_s \leftarrow (t_s - \Delta T_{res}/2, t_s + \Delta T_{res}/2)$
L2	$R \leftarrow \rho(S, G)$
L3	$t_{min} \leftarrow open(\bar{T}_s) + R/v_{max}$

Table 4-5: INIT_SEGMENT(S,G) Algorithm

```
L4    $t_{max} \leftarrow open(\bar{T}_s) + R/v_{min} + \delta t_g$
L5    $\bar{T}_{reach} = (t_{min}, \infty)$, $\bar{T}_{allow} = (-\infty, t_{max})$
L6    $\bar{T}_g \leftarrow \bar{T}_{reach} \cap \bar{T}_{allow}$
L7    $I \leftarrow$ CREATE_ISE(), $init(I) \leftarrow TRUE$
```

The final step in initialization is to create the ISE instance for the path search. Once initialization is complete, the segment is ready for planning. The function PLAN_SINGLE_GOAL(S, G), as defined informally in Table 4-6, begins by defining goals at the goal position but evenly spread over the goal completion time interval, separated by times equal to the DPARMS time equivalence class resolution ΔT_{res} (see lines L10-L13, and Figure 4-6c, where the distributed goals are represented as a string of cells spread across the goal completion time interval). The time interval between goals reflects the DPARMS resolution limit in ISE. Recall that during a search, the ISE resolution pruning mechanism eliminates redundant states according to the objective function and the $better(X, Y)$ function (see Chapter 3.). In general, two goal states in the same DPARMS class would be found to be redundant at the outset. Therefore, goals cannot usefully be spaced more closely than the DPARMS equivalence class resolution.

Table 4-6: PLAN_SINGLE_GOAL(S,G) Algorithm

```
L8    if $init(I) = TRUE$ then
L9        $i \leftarrow 0$
L10       for each time $t \leftarrow open(\bar{T}_g) + i\Delta T_{res}$ such that $t < close(\bar{T}_g)$ do
L11           $g \leftarrow \langle x_g, y_g, t, r_g, c_g \rangle$
L12           $I$:SET_GOAL$(g)$
L13           $i \leftarrow i + 1$
L14       $init(I) \leftarrow FALSE$
L15   $s \leftarrow \langle x_s, y_s, \bar{T}_s, r_s \rangle$
L16   $\langle c^*, s^* \rangle \leftarrow I$:GET_PATH_COST$(s)$
L17   if $c^* = NOPATH$ then return $NULL$
L18   else return GET_PATH$(c^*, s^*)$
```

Once ISE is supplied with goal states, PLAN_SINGLE_GOAL(S, G) makes a single query to ISE using the function GET_PATH_COST(S) with the start state and start time interval (Table 4-6, lines L15-L16). This query calls on either the BESTPCOST or BESTDPARMS modes to find the optimal path from the start to one of the time-distributed goals. In Figure 4-6, frames d) through f) depict the search. Initially, the goal states are the only states in the ISE

OPEN list. They are expanded using all feasible actions (arcs), thereby expanding the search graph. If a plan exists, the graph eventually intersects the start state. Under the BESTPCOST mode, the first detected feasible path will also be the optimal. Under BESTDPARMS mode, ISE finds the "best" feasible state that falls below the path cost maximum. $\mathrm{GET\_PATH\_COST}(S)$ returns the cost of the optimal path $c*$, along with the initial state $s*$. If a path was found, the TEMPEST function $\mathrm{GET\_PATH}(c*, s*)$ yields the plan. The plan begins at the start state position, at some time within the start time interval, and arrives at one of the goals in the goal window after completing the goal action (Figure 4-6f). Note how the DPARMS time class resolution limits the accuracy and precision of the plan - the accuracy in terms of the proximity of the plan start time to the actual start time, and the precision in terms of the coarseness of the goal completion times.

4.3.3 Single-Goal Re-Planning

TEMPEST enables two types of re-planning - *state update re-planning* and *model update re-planning*. Invariably, mission execution does not follow plans precisely. Often, unforeseen operational events cause deviations from the route, schedule or resource guidelines. In response to these deviations, a user calls $\mathrm{PLAN\_SINGLE\_GOAL}(S', G)$ with the updated state. Since the segment goals are initialized, the function skips the goal setting steps and simply re-queries ISE with the updated initial rover state (L16). ISE extends its graph to the new state, yielding a new optimal plan. There is no guarantee that the updated solution is similar to the original. This is state update re-planning.

Re-planning must also occur when changes in the *World Model* or *Rover Model* alter arc transitions, either in terms of state change or arc cost, as encoded through APARMS. Given model updates, TEMPEST reports the APARMS changes for each affected IPARMS state set to the ISE instance, and then calls $\mathrm{PLAN\_SINGLE\_GOAL}(S, G)$. ISE determines the states affected by the updates in $\mathrm{GET\_PATH\_COST}(S')$ and repairs the nodes in the graph to reflect the new optimum. ISE, and hence TEMPEST, is efficient because it restricts its computations to the set of nodes affected by the changes.

Typically, new data about the world comes from rover sensors. Under the backwards-chaining search, the rover position is at a leaf node of the search graph. Therefore, local changes deriving from rover sensor data affect only the *ends* of the graph, and are inexpensive. In contrast, global *World Model* changes and basic changes to the *Rover Model* often affect a large portion of the search graph, and can lead to far more expensive searches (see Chapter 3. for experimental results).

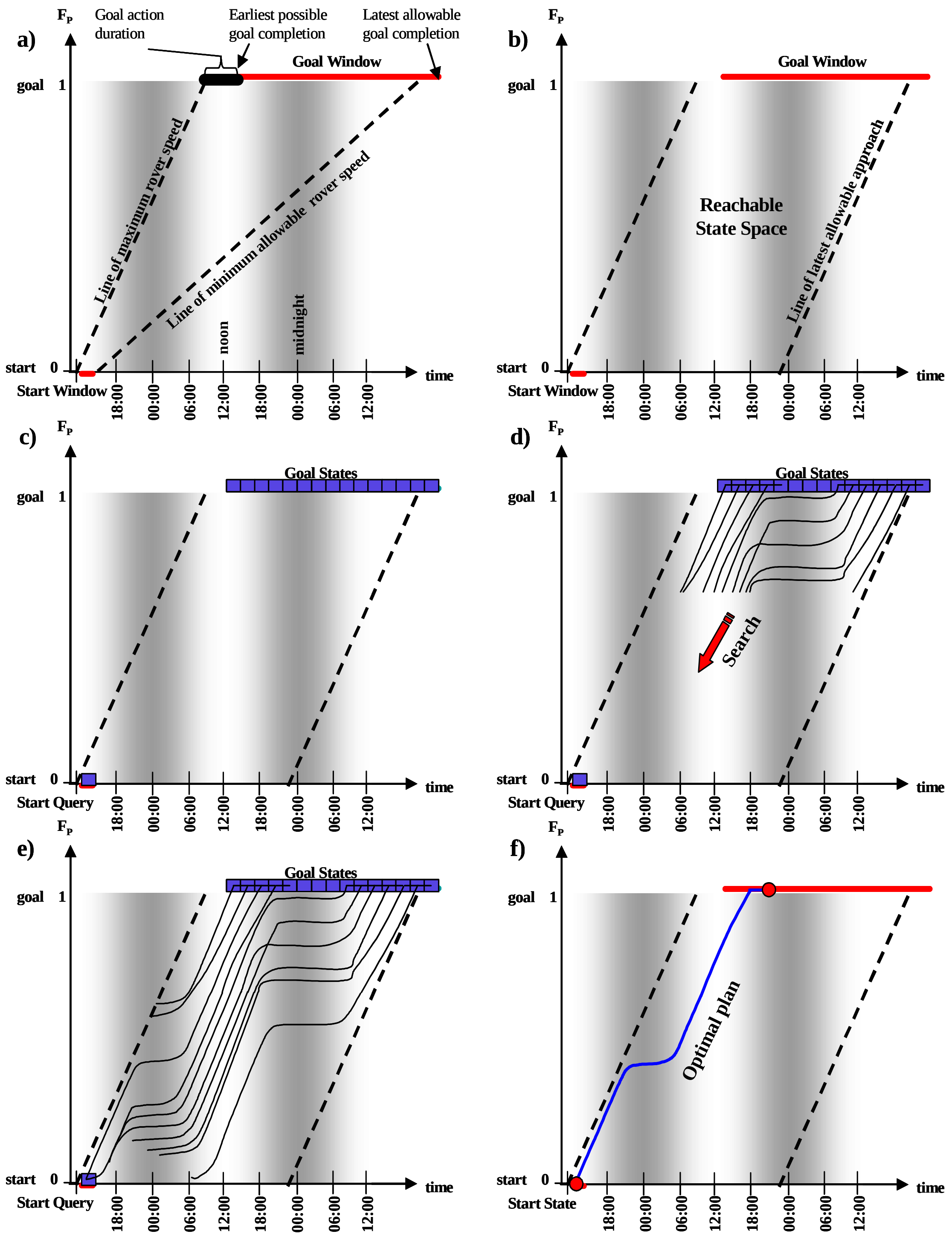


Figure 4-6: Single-Goal Planning. a) Goal time interval definition; b) Reachable and allowable space; c) Goal states and start query; d) Progression of search from goal states; e) Completion of search; f) Optimal plan

4.3.4 Sequential Goal Planning

The TEMPEST sequential goal planning problem involves planning a path from a start state S through a sequence of goals $\Gamma = G_1, \ldots, G_N$ (see Figure 4-7). In general, each goal is position-referenced action that must be executed before moving to the next goal. Each successive pair of goals in the sequence defines a segment. By chaining ISE searches in series, TEMPEST enables planning through a sequence of goals. TEMPEST creates a separate ISE instance for each segment. As with ISE search, TEMPEST sequential goal planning happens in backwards chaining order, from the last segment to first segment.

Sequential goal planning is similar to single goal planning, but differs in some important ways. Principally, the optimal solution for an arbitrary segment is not necessarily part of the optimal solution for the entire goal list. Therefore, it is incorrect to simply chain locally-optimal segments to form the globally-optimal solution. Instead, TEMPEST tracks multiple path candidates through all the segments, then solves for the optimal surviving candidate in the first segment. We describe the process below.

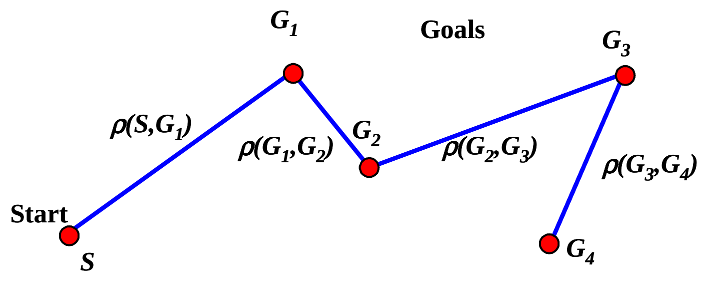


Figure 4-7: The Sequential Goal Planning Problem

As with single-goal planing, TEMPEST begins by initializing the start and goal time intervals for all segments. This requires a new function, $\text{INIT\_SEGMENTS}(S, \Gamma, k)$, whose algorithm appears in Table 4-5. Figure 4-8 contains diagrams depicting the essential steps of sequential goal segment initialization and planning. As in Figure 4-6, the horizontal axes represent time. The vertical axes span all the goals, whose vertical separation corresponds to the distance between them. Each vertical interval between goals represents one of the “segments” of the total path. As with single-goal planning, the process for segment planning proceeds in backwards-chaining order, from the top right of the diagram to the bottom left.

Rather than initializing a single segment, the new function initializes the range of segments, in reverse order, from some final segment k, and through an earliest segment defined by $next(\Gamma)$. In the first call to the function, $next(\Gamma)$ is equal to 1 and k is given a value of N, the total number of segments. For re-planning, the limits take on different values.

Lines L21 through L24 define the minimum and maximum times for the final goal. The minimum distance to the final goal is now the sum over all minimum segment distances (L22). The minimum possible time is calculated in the identical way to single-goal planning. The maximum allowable time now takes into account the sum of goal action durations. This calculation is depicted graphically in Figure 4-8a). The line of fastest approach rises steeply from the opening of the start time interval, while the line of slowest allowable approach ascends more slowly, with pauses for each goal action. As before, these times define the *reachable* and *allowable* time ranges for the goal, whose set intersection becomes the final goal time interval.

Table 4-7: INIT_SEGMENTS(S,Γ,k) Algorithm

```
L19  $\bar{T}_s \leftarrow (t_s - \Delta T_{res}/2, t_s + \Delta T_{res}/2)$
L20  for each goal $i \leftarrow k, \ldots, next(\Gamma)$ do
L21      if $i = N$ then
L22          $R \leftarrow \rho(S, G_{next(\Gamma)}) + \sum_{j = next(\Gamma)}^{N-1} \rho(G_j, G_{j+1})$
L23          $t_{min} \leftarrow open(\bar{T}_s) + R/v_{max}$
L24          $t_{max} \leftarrow close(\bar{T}_s) + R/v_{min} + \sum_{j = next(\Gamma)}^{N} \delta t_j$
L25      else
L26          $t_{min} \leftarrow open(\bar{T}_{i+1}) - \rho(G_i, G_{i+1})/v_{max}$
L27          $t_{max} \leftarrow close(\bar{T}_{i+1}) - \rho(G_i, G_{i+1})/v_{max} - \delta t_{i+1}$
L28      $\bar{T}_{reach} \leftarrow (t_{min}, \infty)$ , $\bar{T}_{allow} \leftarrow (-\infty, t_{max})$
L29      $\bar{T}_i \leftarrow \bar{T}_{reach} \cap \bar{T}_{allow}$
L30      $I_i \leftarrow$ CREATE_ISE() , $init(I_i) \leftarrow TRUE$
```

The earlier segment goal time bounds are computed recursively from later goal time intervals. The first reachable time is computed by subtracting the minimum traverse time from the *opening* of the next goal interval (L26). The last allowable time is found by subtracting the fastest possible traverse and the duration of the next goal's action from the *closing* of the next goal interval (L27). As with the final goal, an earlier goal time interval is the intersection of the reachable and allowable time bounds (L29). Figure 4-8b) shows the calculation graphically. From the final goal time interval, the line defining the earliest reachable bounds extends downwards without pause from the opening of the final goal time interval, while the boundary defining the latest allowable times descend in stair-step fashion, pausing for each goal action. The function creates an ISE instance for each segment, and returns upon initializing the first remaining segment, given by $next(\Gamma)$.

Once the segments are initialized, planning occurs in an updated function $\mathrm{PLAN\_SEQ\_GOALS}(S, \Gamma)$, defined in Table 4-8, and depicted in the remainder of Figure 4-8. Starting with the last unplanned segment, the function alternates between defining the ISE goal states and planning the segment. If the last unplanned segment is also the final segment, the goals are defined as in single-goal planning (L36-L39). Before moving on to earlier segments, the function proceeds to planning for the current segment. Recall that the locally optimal segment solution is not, in general, part of the global optimum. To account for this, TEMPEST generates a number of time-distributed plan solutions. First, it defines a start time interval for the segment, equal to the previous goal time interval (for all but the first segment - see L46), and equal to the overall plan start time interval for the first segment (L48). At regular intervals within the start time interval (the time DPARMS resolution equivalence resolution), $\mathrm{PLAN\_SEQ\_GOALS}(S, \Gamma)$ makes a query to ISE with $\mathrm{GET\_PATH\_COST}(S)$ (L49-L52). If the query results in a solution, the solution is recorded. If not, the function continues with next query. Figure 4-8c) shows the start state queries as discrete intervals over the start of the last segment. Figure 4-8d) shows how some of these queries result in feasible plan segments, while others do not[1]. Once all the query states are exhausted, the function continues on to the next earliest segment. However, if no solutions result from the queries, then there is no feasible plan for the problem.

To define the goals for earlier segments, the function copies the initial states and path costs from the following segment's path solutions (L41-L43). Again, planning proceeds as a series of queries to an ISE planner, and may result in a list of plan segment solutions. The process repeats for all segments down to the next remaining segment (Figure 4-8e). The effect is to chain segment solutions together to build long, consistent plans one segment at a time. The earliest segment has only one valid start time interval, and so involves only one ISE query. The resulting solution is the global optimum for the sequential goal problem, depicted in Figure 4-8f).

4.3.5 Sequential Goal Re-Planning

TEMPEST also provides re-planning for multi-goal traverses. This contrasts with single-goal re-planning in that model changes may affect more than one segment, and hence more than one ISE graph. TEMPEST must notify each affected ISE instance, but need only initiate re-planning from the latest affected segment. Note the analogy to modifications in a single ISE search - updates near the position of the rover tend to affect fewer segments than updates near the final goal, and hence are far cheaper to re-plan.

1. Note that some goal states and start states do not produce feasible solutions. Further note that each start state can only have one goal state (the optimal path is unique, barring ties), but that the optimal paths from several start states may all arrive at the same goal state.

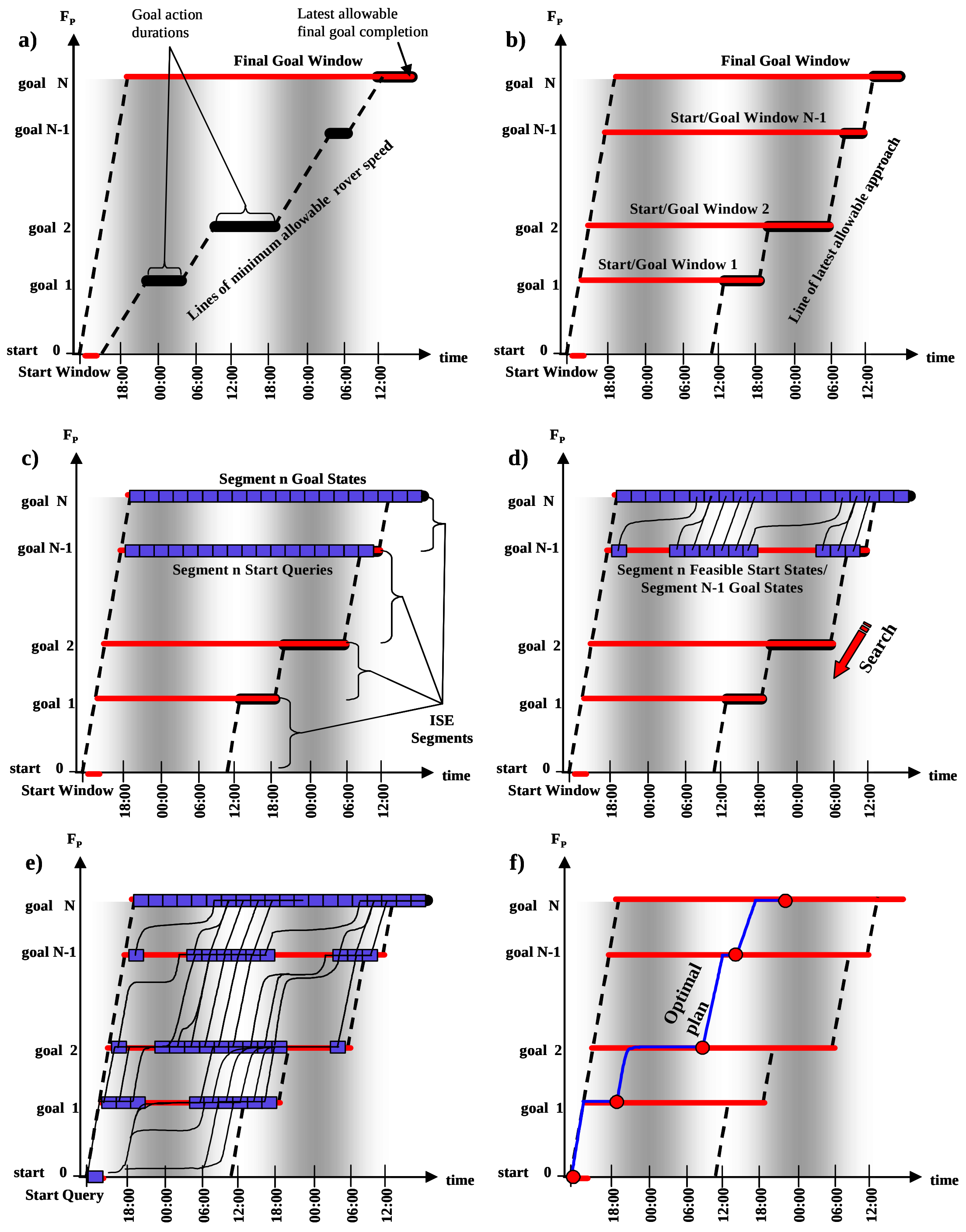


Figure 4-8: Sequential Goal Planning. a) Final goal time interval definition; b) Previous segment goal time interval definitions; c) Segment goal states and start queries; d) Planning in last segment and generation of goal states for previous segment; e) Completion of planning; f) Optimal plan

An update procedure precedes re-planning. TEMPEST determines which segments are affected by updates that occur during plan execution. An update to the start state affects only the first remaining segment. Updates to models may affect one or more segments. Successfully completing a goal or abandoning a goal removes the corresponding segment from consideration in future planning. All segments affected by updates and all segments preceding the affected segments must be re-planned. The reason is that in a backwards search order, the solutions for earlier segments are based on solutions for later segments. Modifications to arc costs in a later segment in general alter the solutions for that segment, and hence the goals for the previous segment. So, a change in a later segment invalidates all earlier goal states. Once TEMPEST determines which segments are affected by updates, it deletes all but the last affected ISE instances, and calls $\mathrm{INIT\_SEGMENTS}(S, \Gamma, k)$, where k is the index of the second-to-last affected segment, or $last(\Gamma) - 1$. Once the segments are re-initialized, a call to $\mathrm{PLAN\_SEQ\_GOALS}(S, \Gamma)$ initiates re-planning. Each ISE instance repairs its search graph and yields plans, if feasible, for queries over the newly-defined start time intervals. The resulting plan, if any, is the optimum considering the new data.

Table 4-8: PLAN_SEQ_GOALS(S,Γ) Algorithm

```
L31  for each segment $i \leftarrow last(\Gamma), \ldots, next(\Gamma)$ do
L32      $S_i{*} \leftarrow NULL$
L33      if $init(I_i) = TRUE$ then
L34          if $i = N$ then
L35              $j \leftarrow 0$
L36              for each time $t \leftarrow open(\bar{T}_N) + j\Delta T_{res}$ such that $t < close(\bar{T}_N)$ do
L37                  $g \leftarrow \; x_N, y_N, t, r_N, c_N$
L38                  $I_N$:SET_GOAL$(g)$
L39                  $j \leftarrow j + 1$
L40          else
L41              for each $\; c*, s* \; \in S_{i+1}{*}$, with $s* = \; x*, y*, t*, r*$
L42                  $g \leftarrow \; x*, y*, t*, r*, c*$
L43                  $I_i$:SET_GOAL$(g)$
L44          $init(I_i) \leftarrow FALSE$
L45      if $i > next(\Gamma)$ then
L46          $\bar{T} \leftarrow \bar{T}_{i-1}$, $x \leftarrow x_i$, $y \leftarrow y_i$, $r \leftarrow r_i$
L47      else
L48          $\bar{T} \leftarrow \bar{T}_s$, $x \leftarrow x_s$, $y \leftarrow y_s$, $r \leftarrow r_s$
```

Table 4-8: PLAN_SEQ_GOALS(S,Γ) Algorithm

```
L49         for each time interval $\bar{T}_{sub} \leftarrow (open(\bar{T}) + j\Delta T_{res}, open(\bar{T}) + (j+1)\Delta T_{res})$ such that $close(\bar{T}_{sub}) \le close(\bar{T})$
L50             $s \leftarrow x, y, \bar{T}_{sub}, r$
L51             $c^*, s^* \leftarrow I_i$:GET_PATH_COST($s$)
L52             if $c^* \ne NOPATH$ then $S_i^* \Leftarrow c^*, s^*$
L53         if $S_i^* = NULL$ return $NULL$
L54     return GET_PATH($c^*, s^*$)
```

4.3.6 Time-Bounded Sequential Goal Planning

Often, legal goal completion is restricted to within fixed temporal bounds. For example, a communications opportunity may only be possible within a specific time range, or a scientific measurement might only be valuable during certain times of day. One approach to planning in these situations is to add constraints to the *Constraint Set* that define legal time ranges, associate those constraints with the goal actions, and perform sequential goal planning as described in the previous sections. Time constraints on goals can often drastically reduce the reachable state space. Forcing the ISE search to discover the reachable states both upstream and downstream of a time-limited goal is a wasteful activity in light of *a priori* knowledge.

Instead, the INIT_SEGMENTS(S, Γ, k) function can be modified to quickly propagate the effect of goal time constraints to earlier and later goals, thereby eliminating unreachable ranges of the goal time intervals that cannot contribute to feasible plans. This new function, INIT_TB_SEGMENTS(S, Γ, k), can substantially improve performance in finding the same optimal solutions as would be found using the earlier initialization algorithm. It is detailed in Table 4-9. A diagram of the new initialization procedure appears in Figure 4-9.

Figure 4-9a shows the same problem layout as in Figure 4-8a, but with the addition of goal time bounds for goal 2 and N-1, specified by the *Mission Specification*, that restrict the legal range of goal completion times. These goal bounds influence the initialization procedure in the following ways: they potentially further limit the closing time of later goal time intervals; they potentially further limit the closing times of earlier goal time intervals; and they directly limit the time interval for their associated goal.

As with INIT_SEGMENTS(S, Γ, k), the new initialization projects the *reachable* time range for the final goal using the maximum possible speed and shortest distance (L58, L59 and L72). It also projects an *allowable* time range for the final goal (t_{max2}) using the minimum allowable speed, the same distance, and the durations of goal actions (L65). Figure 4-8a illustrates the result of these computations. However, the allowable time range may be further limited by the latest upstream mission-imposed goal time bounds. Consider that if an upstream goal is forced to finish earlier than would be limited by unbounded sequential goal planning alone, it could also limit how late downstream goals can be completed, under the minimum allowable speed restriction. It is pointless to plan for goal completion times that can only be reached by slower-than-allowable speeds. Therefore, the algorithm must determine which time limitation is more constraining - the latest allowable time from unbounded sequential goal planning, or the propagated effect of earlier goal time bounds.

Table 4-9: INIT_TB_SEGMENTS(S,Γ,k) Algorithm

```
L55  $\bar{T}_s \leftarrow (t_s - \Delta T_{res}/2,\ t_s + \Delta T_{res}/2)$
L56  for each goal $i \leftarrow k, \ldots, next(\Gamma)$ do
L57      if $i = k$ then
L58          $R \leftarrow \rho(S, G_{next(\Gamma)}) + \sum_{j = next(\Gamma)}^{k-1} \rho(G_j, G_{j+1})$
L59          $t_{min} \leftarrow open(\bar{T}_s) + R/v_{max}$
L60          if $i = N$ then
L61              $t_{max1} \leftarrow \infty$
L62              for each goal $j \leftarrow N, \ldots, next(\Gamma)$ do
L63                  if $close(\bar{T}_{Bj}) \neq \infty$ then
L64                      $t_{max1} \leftarrow close(\bar{T}_{BJ}) + \frac{1}{v_{min}} \sum_{m=j}^{N-1} \rho(G_m, G_{m+1}) + \sum_{m=j+1}^{N} \delta t_m$, break
L65              $t_{max2} \leftarrow close(\bar{T}_s) + R/v_{min} + \sum_{m = next(\Gamma)}^{N} \delta t_m$
L66              $t_{max} \leftarrow min(t_{max1}, t_{max2})$
L67          else
L68              $t_{max} \leftarrow close(\bar{T}_{i+1}) - \rho(G_i, G_{i+1}) - \delta t_{i+1}$
L69      else
L70          $t_{min} \leftarrow open(\bar{T}_{reach}) - \rho(G_i, G_{i+1})/v_{max}$
L71          $t_{max} \leftarrow close(\bar{T}_{i+1}) - \rho(G_i, G_{i+1})/v_{max} - \delta t_{i+1}$
L72      $\bar{T}_{reach} \leftarrow (t_{min}, \infty)$
L73      $\bar{T}_{allow} \leftarrow (-\infty, t_{max})$
L74      $\bar{T}_i \leftarrow \bar{T}_{reach} \cap \bar{T}_{allow} \cap \bar{T}_{Bi}$
```

Table 4-9: INIT_TB_SEGMENTS(S,Γ,k) Algorithm

```
L75        if $\bar{T}_i = NULL$ then return $FALSE$
L76        else $I_i \leftarrow$ CREATE_ISE() , $init(I_i) \leftarrow TRUE$
L77    return $TRUE$
```

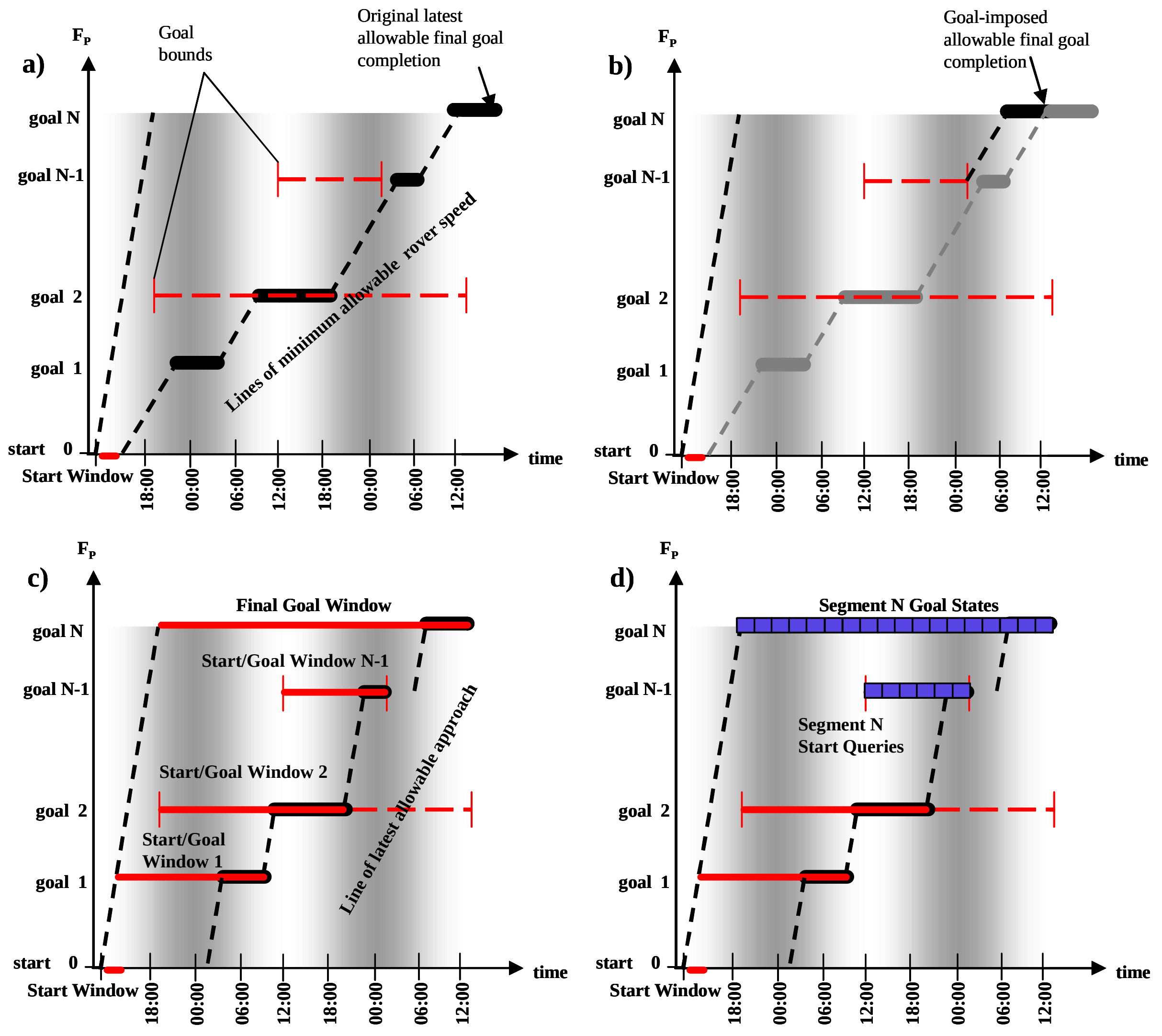


Figure 4-9: Initialization for Time-Bounded Sequential Goal Planning: a) Originally-specified reachable and allowable time limits; b) Earlier goal-imposed allowable time limit; c) Goal time interval definitions; d) Final segment goal states and query start states

Figure 4-8b shows that the time bound for goal N-1 limits the goal interval for goal N. The projection of the slowest allowable speed for segment N, summed with the goal N action duration, is earlier than the end of the original allowable time interval. Any arrival time later than this new time would require an approach to goal N by a slower-than-allowable speed.

Returning back to Table 4-9, in computing the final goal allowable time interval, the algorithm searches to find the last time-bounded goal in the goal sequence (L62 and L63). It projects the latest allowable time from the end of that goal time bounds to the final goal (t_{max1}, from L64), and keeps the minimum of t_{max1} and t_{max2} as the true closing time (L66 and L73). For earlier start intervals, the time bounds propagate from the next latest goal interval just as with $\text{INIT\_SEGMENTS}(S, \Gamma, k)$ (L70 and L71). The only difference is that the overall goal time interval is the intersection of three time ranges - the reachable, allowable and goal-limited time bounds.

Figure 4-8c shows the effect of the intersection. In the specific case of goal N-1, the latest allowable time falls *later* than the mission-imposed goal time bounds. The intersection removes a large time interval from the allowable unbounded sequential goal range from consideration. In the case of goal 2, the latest allowable time falls *earlier* than the mission-imposed bounds, causing the minimum allowable speed restriction to take precedence. For goal 1, no mission-imposed time bounds exist, and so again, the minimum allowable speed restriction takes precedence. Importantly, if at any time during the initialization a goal time interval is computed to be empty, the mission, as specified, is infeasible.

The new initialization procedure is all that is required to enable TEMPEST to plan and re-plan for time-bounded sequential goals. Once segments are initialized, either prior to planning or in response to execution updates, $\text{PLAN\_SEQ\_GOALS}(S, \Gamma)$ solves for the plan that obeys the temporal bounds on goal completion.

4.4 Plans

TEMPEST plans consist of a fully-ordered sequence of state-action pairs, called *waypoints*, corresponding to the states and arcs that follow the optimal path from the start state, through all intermediate goals, to one of the designated final goal states. A plan π with N waypoints takes the form:

$$\pi \equiv \{w_1, w_2, ..., w_N\} \qquad 4\text{-}1$$

where each waypoint takes the following form, and is interpreted as follows:

$$w_i \equiv \{x_i, y_i, t_i, e_i^1, \ldots e_i^J, a_i\} \quad \text{where} \quad \begin{array}{l} x_i \equiv \text{x-coordinate of map cell} \\ y_i \equiv \text{y-coordinate of map cell} \\ t_i \equiv \text{ arrival ephemeris time} \\ e^j{}_i \equiv \text{minimum allowable level of jth resource} \\ a_i \equiv \text{action starting at } t_i \end{array} \qquad \text{4-2}$$

Each state is an aggregation of the ISE IPARMS, DPARMS and auxiliary resource variables for the application. The corresponding action is the action departing from the state.

As a sequence of waypoints, a plan acts as a guide for path execution whose granularity is defined by the resolution of the spatial coordinates and the other state variables. Each action in the plan must be initiated as close as possible to the position and time of the waypoint. Each resource variable specifies the minimum resource level that must remain in order to satisfy all the goals in the plan, as described in Chapter 3, Section 3.1.5. A deviation from plan position greater than the spatial resolution of the map, or in timing greater than the time equivalence class resolution may justify re-planning. Similarly, if any resource falls below the recommended minimum level as dictated by the plan, successful re-planning is necessary to ensure mission completion.

4.5 Plan Evaluation

In evaluating TEMPEST planning behaviors, it is useful to introduce some approaches for analyzing plans. Plan solutions are often difficult to interpret. Even when a planner produces formally correct and optimal plans, it is not immediately obvious whether plans display a desired behavior, how much of a behavior to attribute to artifacts of representation versus to good planning, or how to compare plans stemming from alternate approaches. This section presents some simple analysis tools that help with this problem.

4.5.1 Distance

Plan distance is an important path planning measure. For many path planners, minimizing feasible path length is the single objective. Though minimizing path length is not the only objective for TEMPEST, it is still very important. In many cases, the shortest path is the least costly in terms of resources, mechanical stress and vehicle risk. This thesis introduces two factors that describe a plan’s increase in path length above the planar Euclidean distance between goals.

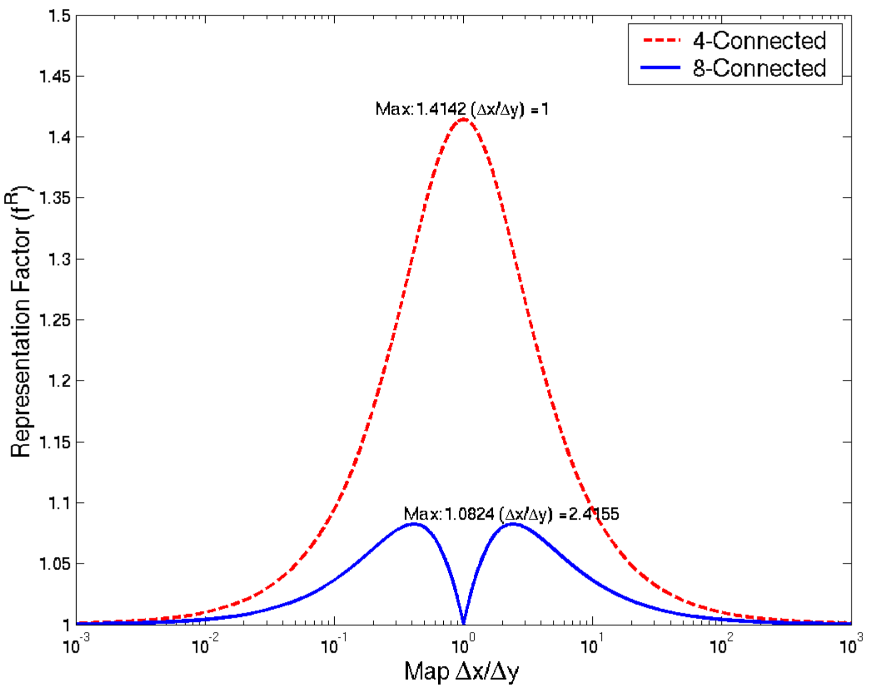


Figure 4-10: Representation Factor for Regular Four and Eight-Connected Grids

Representation Factor f_R : This factor encodes the ratio of the minimum distance possible under the planner's state representation (D_{rep}) to the planar Euclidean map distance (D_{map}) such that:

$$f_R = D_{rep}/D_{map} \qquad \text{4-3}$$

Representation factor encodes how much a planner's underlying spatial representation contributes to an extension of path length beyond the minimum. Representation factor has a minimum value of 1, and a maximum value that depends on the system of spatial representation. TEMPEST uses a grid-based representation for terrain. Actions transition between cells in the grid and their eight nearest neighbors, forming an eight-connected graph. The minimum eight-connected plan distance between two points depends on the ratio of their relative grid distance in the x-coordinate *(Δx)* and the y-coordinate *(Δy)*. When the start and goal lie along a common horizontal (*Δx/Δy*=∞), vertical (*Δx/Δy*=0) or principle diagonal axis of the graph (*Δx/Δy*=1 or *Δx/Δy*=-1), the minimum eight-connected distance is equal to the Euclidean map distance between the points ($f_R = 1$). Since the eight-connected path cannot assume arbitrary headings, other ratios of Δx-to-Δy produce indirect paths whose path lengths exceed the Euclidean distance between the points ($f_R > 1$). Figure 4-10 plots f_R for both four- and eight-connected motion on regular grids given the relative positioning of the start and goal. The four-connected graph, corresponding to the "Manhattan distance", yields a worst-case error of $\sqrt{2}$ for two points directly on a diagonal. At worst, the eight-connected graph contributes a 8.2% increase in path length.

Avoidance Factor f_A: This factor encodes the ratio of the plan distance (D_{plan}) to the minimum distance possible under the planner's state representation (D_{rep}) such that:

$$f_A = D_{plan}/D_{rep} \qquad 4\text{-}4$$

Avoidance factor captures the amount of extra distance, inserted by a planner, to avoid obstacles and sub-optimal regions. Its minimum value is one, and its maximum value is unbounded. It is important to note that an avoidance factor of 1 does not mean that the path it describes has not avoided obstacles or areas of high cost. In general, a path with the minimum representation distance D_{rep} is not unique, and has many degrees of freedom with which to avoid obstacles or high cost areas. Observe in Figure 4-11 that the solid paths are all of minimum length under an eight-connected grid representation $f_A = 1$, and yet avoid obstacles. The dashed path, however, covers more distance, and so has an avoidance factor greater than one.

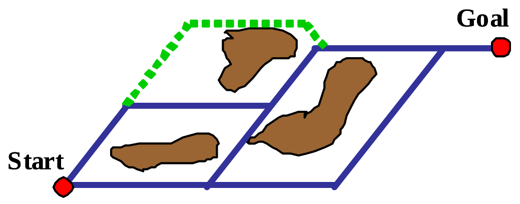


Figure 4-11: Obstacle Avoidance with f_A=1 (solid) and f_A>1 (dashed)

For plans solved under an objective function that minimizes path length, avoidance factor will be greater than 1 only if obstacles prevent all paths yielding the minimum representation distance D_{rep} between points. For objective functions that minimize some other quantity, like duration or energy expense, an avoidance factor greater than one may indicate a deviation in path to avoid a costly region.

4.5.2 Time

In temporal planning, where actions are not necessarily mobile, plan duration is a more appropriate measure of plan length than distance. For mobile actions, the distance factors in Section 4.5.1 also encode the increase in plan duration due to increased path length. Beyond this, stationary actions inserted into a plan also cause an increase in plan duration. If the minimum map traverse time and total duration of goal actions are defined respectively as:

$$\Delta T_{min} = D_{map} / v_{max} \qquad 4\text{-}5 \qquad\qquad \Delta T_{goal} = \sum_{i=1}^{N} \delta t_i \qquad 4\text{-}6$$

then the minimum required plan duration is given by:

$$\Delta T_{req} = \Delta T_{min} f_R f_A + \Delta T_{goal} \qquad 4\text{-}7$$

Slower driving speed and additional stationary actions inserted into the plan motivate the following additional factor:

Loiter Factor f_L **:** This factor encodes the ratio of the minimum plan duration, considering traverse time and goal action durations, to the actual plan duration, such that:

$$f_L = \Delta T_{plan} / \Delta T_{req} \qquad 4\text{-}8$$

Loiter factor expresses the degree to which a plan specifies less-than-maximum driving speed or stationary actions beyond the goal actions.

4.6 Simulation Results

4.6.1 Re-Planning

We present a sample planning problem to illustrate TEMPEST behaviors. A contour map in Figure 4-12 shows the elevation profile for synthesized terrain on a mock Martian surface. Mountains form a central pattern of valleys running in a North-South (up-down) direction, and a rounded crater lies to the northeast. The rover starts in the morning at the southeast corner of the map (“Start”), and must traverse to the northwest corner (“Goal 2”). Scientists designate a Via Point goal in the valley, “Goal 1”, to promote travel through the valley en route to the final destination. Mission engineers (or an onboard planner-scheduler) require the robot to reach Goal 2 with 100 W-hr of charge left for subsequent operations.

Unfortunately, the elevation map provided to the rover is incorrect. Its preliminary map indicates a clear exit from the valley system at its northern extreme, between two peaks. The actual terrain involves a far higher and steeper pass, beyond the locomotion capability of the rover.

In this example, a simple simulation of the mission demonstrates the value of TEMPEST. At each plan step, the simulated rover plans a path from its current state, then "executes" the first step of the plan through path integration. At each new point, the rover "senses" the local environment, detecting the actual elevation, slope and lighting of all cells within two pixels. Based on this new data, TEMPEST re-plans a path and the process continues.

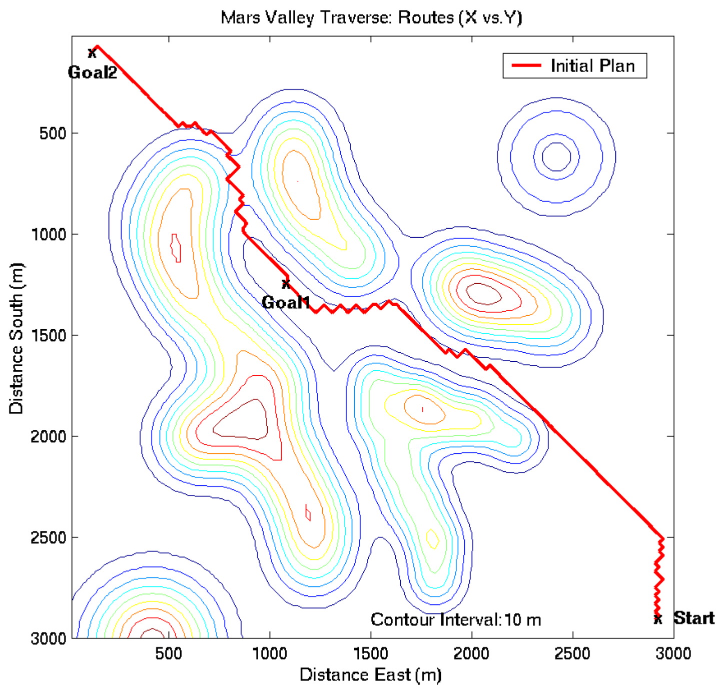


Figure 4-12: Initial Plan Route: TEMPEST plans a path from "Start" at the southeast of the map, to Goal 1 in the valley in the center of the map, and then through the saddle point to Goal 2 in the northwest.

Figure 4-12 shows the *initial* TEMPEST plan route. With the exception of a few minor path deviations, the route follows a direct path from the start through each of the goals. Subsequent re-plans during "execution" yield similar routes. The solid red curve in Figure 4-14 shows the timing for the initial plan. The slope of the curve represents the rover speed toward Goal 2. One observes that it is only slightly slower than the theoretical fastest, straight-line approach, as shown by the steep dash-dot line. The red solid line in Figure 4-15 shows the required battery energy profile for the initial plan. The plan allows the robot to begin with an empty battery, and only requires increasing charge at the end of the plan to meet the Goal 2 requirement. This indicates that solar energy provides ample energy for locomotion.

The simulated rover reaches the Goal 1 Via Point and continues without stopping toward Goal 2. The nearest valley exit, according to its internal elevation map, lies to the northwest directly in line with its next goal. However, as it approaches the supposed low pass, the robot detects the steep, intraversible pass. Figure 4-13 shows the first substantial re-plan in the sequence, based on this discovery. With no escape to the northwest, TEMPEST selects the least expensive alternative - a detour through a narrow valley to the northeast (the blue dashed line). This new route is a significant departure from the original. The extra distance means that the robot cannot reach Goal 2 before sundown.

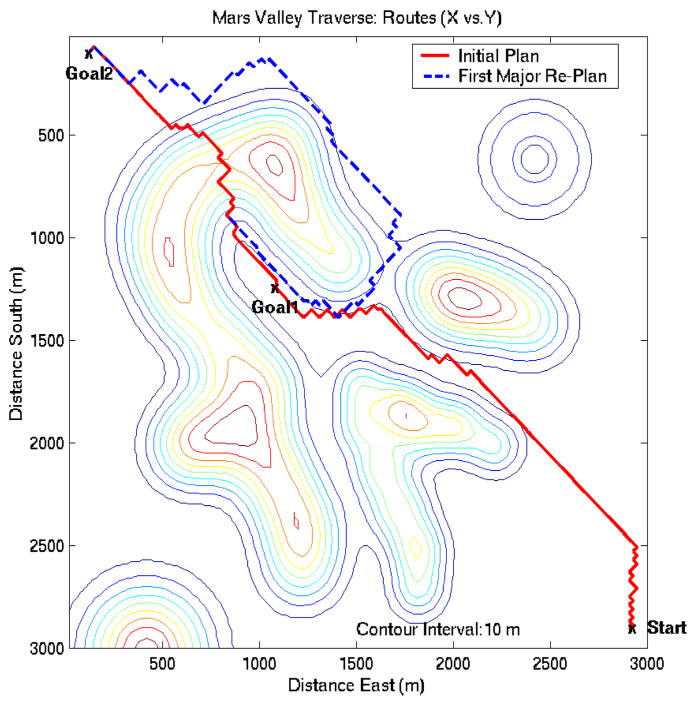


Figure 4-13: First Significant Re-Plan Route: The robot discovers a much steeper approach to the saddle point at the end of the valley after visiting Goal 1, prompting a detour through an opening to the valley to the northeast.

The original plan did not anticipate the extra burden of nightfall on battery reserves. However, TEMPEST determines a prolonged *Charge* action followed by overnight *Hibernation* will enable it to reach Goal 2 by late morning the following day. Figure 4-14 shows the rate of travel towards Goal 2 for the detour as a dashed blue line. Note that the robot must first reverse course, and then remains stationary for nearly 18 hours, first in sunlight (charging batteries), then overnight (hibernating, in the shaded region). The following morning, the rover continues its course around the mountain, then moves to Goal 2.

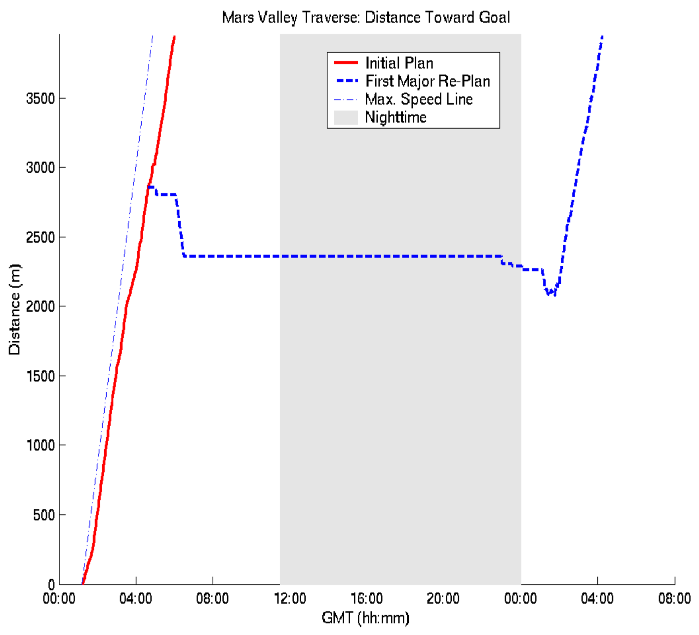


Figure 4-14: Progress Distance vs. Time: The initial plan (shown in red) follows a very direct path (compare to the straight-line maximum speed curve). The re-plan detour (shown in blue) requires the rover to endure a night in hibernation, as shown by the flat region indicating no forward progress. In the morning of the following day, the rover resumes its course to Goal 2.

Figure 4-15 shows the required battery energy profile over the same time span, also with a blue dashed line. Note that the re-plan still enables the robot to start from an empty battery. However, well in advance of nightfall, the plan requires a steady increase in battery charge to generate reserves for the night. The battery energy requirement falls overnight - the morning sun is sufficient to charge the battery to the required Goal 2 level.

Standard path planners like A* or D* do not adequately address the problem presented in this example. While A* can efficiently find a path to avoid terrain obstacles, and D* is able to re-plan efficiently to avoid unexpectedly difficult terrain, neither is able to anticipate the need to charge in advance of nightfall. Depending on the actual initial battery state-of-charge of the rover, following a plan that ignores energy could have disastrous consequences. The example can be extrapolated to the consideration of other resources, which might also be critical to data quality, time efficiency or rover survival.

.

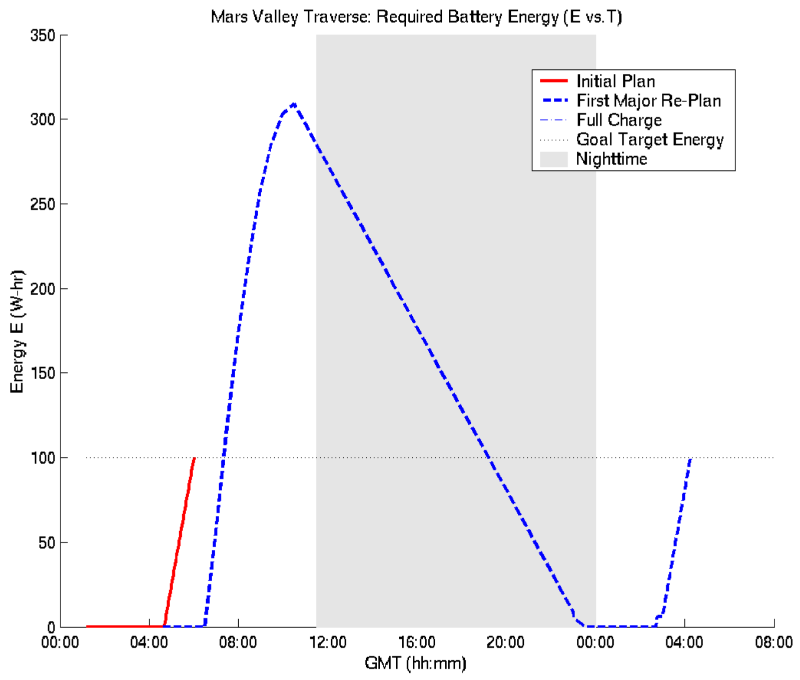


Figure 4-15: Battery Energy Requirement vs. Time: The initial plan enables the rover to start from total battery discharge to reach to the goal charge level. The detour from the re-plan requires the rover to perform stationary charging to nearly full capacity in anticipation of the nighttime hibernation. Note the similarity between the re-plan profile in the morning after hibernation and the original plan profile.

4.7 Discussion

This chapter presents an approach to mission-directed path planning that displays elements of the five attributes listed in Chapter 1. It displays over-the-horizon foresight by considering large-scale terrain beyond the view of the robot in planning. It exhibits temporal and resource cognizance through a consideration of time and energy variables in the state space, and in the enforcement of resource capacity constraints through ISE global constraint planning. It demonstrates an ability to handle some degree of uncertainty by enabling efficient re-planning in response to state and model updates. And the approach directs its focus to the mission objectives in terms of achieving goals, accommodating goal action requirements and respecting constraints on activities.

The example in simulation highlights how TEMPEST coordinates route, timing and battery energy to achieve multiple goals. Unlike many approaches to temporal planning, TEMPEST approaches the problem as a whole rather than by a simpler, but sub-optimal, hierarchical breakdown. Incorporating this strategic planning capability into a rover could provide a significant boost to rover safety, mission time efficiency and reliability. Specifically, TEMPEST enabled a contingency plan that became necessary once the robot detected that its original route was not feasible. Where a rover employing traditional techniques would have had to suspend its operations to wait for further instructions from human operators, the simulated robot was able to re-specify its route and schedule to achieve the originally-stated objectives.

The examples in this thesis all deal with problems combining space, time and energy. Other factors are important to planetary surface exploration and might be considered under the TEMPEST approach. Rover health, limited by rough driving, extended exposure to dust, or thermal cycling, could be treated as a resource. One or more rover auxiliary health variables could be added to a model. The state transition function, as derived from the *World Model* and *Rover Model*, could describe the effect of activities on vehicle health. Greater mechanical damage might be sustained in areas of rough terrain or on steep slopes. Dust might accumulate on solar arrays as a function of time[1], and, if represented as a DPARMS state variable, could adversely affect the collection of solar energy. The transition function could model the change in temperature of certain components as a function of activity level, sun exposure and material properties. Thermal cycles of sufficient size might accumulate in another counter variable. TEMPEST could plan paths that prevent vehicle health variables from dropping below minimum tolerable lower bounds. Alternatively, one type of vehicle damage cost could be minimized over a path, while meeting constraints on other variables and mission goals.

Data from science and exploration activities might also be considered a resource. Data transmitted to Earth might be a resource to maximize. Completion of goal activities could add data to vehicle memory. Limitations in the size of memory could limit the data stored aboard the rover. Communications activities could downlink data to Earth, at a maximum data rate, to achieve greater overall reward. Downlink could only happen when in view of Earth or an orbiting relay spacecraft.

1. However, a monotonic increase in dust is probably inappropriate for Mars. Both Spirit and Opportunity experienced several events that removed a substantial layer of dust from their arrays, presumably from Martian "dust devils."

5. Sun-Synchronous Navigation

Sun-synchronous navigation is a promising strategy for rover-based planetary exploration in polar environments. It entails synchronizing a robot's route and timing with the motion of the sun to provide continual solar energy while maintaining a benign thermal environment. In 2001, the Sun-Synchronous Navigation project built a solar-powered rover, Hyperion, and software to achieve semi-autonomous sun-synchronous navigation, and tested the combined system in planetary-relevant polar terrain on Devon Island in the Canadian Arctic. A significant outcome of that research effort was the first version of TEMPEST, tailored specifically for sun-synchronous route planning. As a background to the development and testing of TEMPEST, this chapter begins by describing the sun-synchronous navigation strategy and provides an overview of the field experiment, the rover Hyperion, and its software architecture. Several following sections then describe the planning approach used, and provide experimental results from the Arctic field experiment.

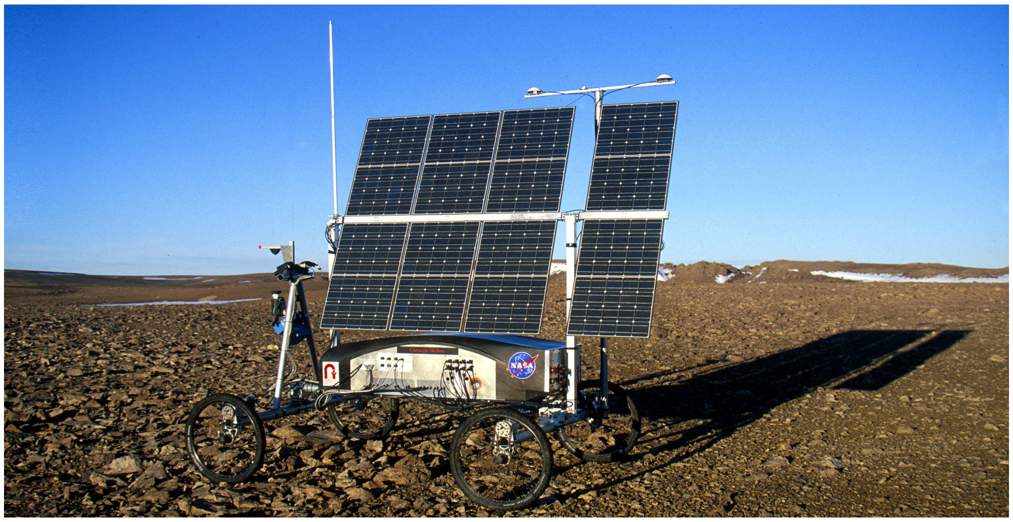


Figure 5-1: Hyperion Rover: Sun-synchronous navigation enabled solar powered travel over kilometers without the added complication and mass of a gimbal mechanism.

5.1 The Polar Navigation Problem

The idea for sun-synchronous navigation emerged from mission design studies in the context of lunar polar exploration. The Moon's axis of rotation is oriented just 1.53° from perpendicular to the ecliptic plane, the plane of the Earth's orbit about the Sun [23]. Therefore, directly at either of the lunar poles, the maximum sun elevation angle ever achieved is 1.53°. In the summer months, as on Earth's poles, the sun remains above the horizon continually, albeit only above the lunar "arctic circles" at 88.47° latitude, within just 46 km from the poles. But with such low-skimming sun angles, terrain features cause extensive shadows, decreasing the effectiveness of solar-powered operations. Furthermore, without an atmosphere, the thermal contrast between sunlit and shadowed conditions is drastic.

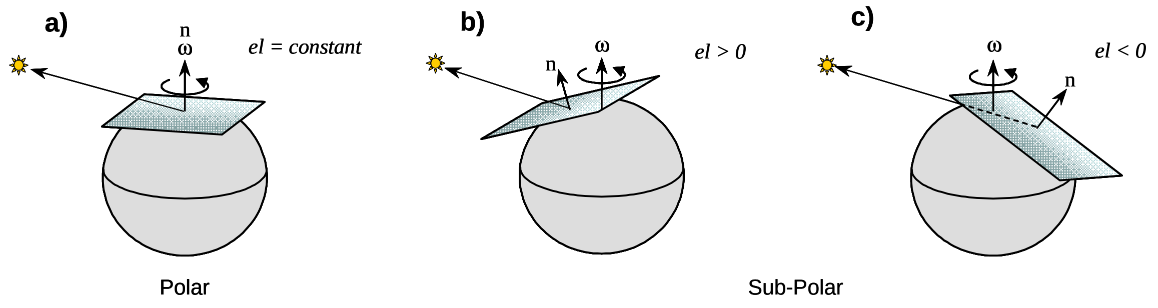


Figure 5-2: Lunar Sun Elevation Angle Variations for Polar and Sub-Polar Positions. At mid-summer at a pole (a), the sun remains above the horizon at a roughly constant angle. Below the arctic circle, the sun elevation oscillates between a maximum elevation (b) above the horizon and a minimum elevation (c) below the horizon.

One might suspect that moving slightly away from the poles would improve sunlight exposure. Figure 5-2 is a diagram showing sun elevation angles for polar and sub-polar positions. It depicts the Moon's axis of rotation ω and observing positions with surface normals *n*, which coincide with the zenith vectors. Directly at a lunar pole in summer, the sun elevation is roughly constant, inscribing a "halo" in the lunar sky, just above the horizon, whose center is at zenith (Figure 5-2a). Moving away from the poles to lower latitudes, the zenith vector oscillates around the pole, nodding towards and away from the sun. The center of the "halo" inscribed by the sun in the lunar sky tilts a corresponding angle from the zenith, causing the sun elevation to oscillate between a minimum value at local midnight and a maximum value at local noon. At the arctic circle, the sun meets the horizon at its lowest point in the sky, and past the arctic circle towards the equator, the sun dips below the horizon for some portion of each lunar rotation (Figure 5-2c). Meanwhile, with increasing distance from the pole, the peak elevation continues to increase (Figure 5-2b). So, by moving from the pole, sun angles get better *and* worse from a solar power standpoint - higher mid-day sun angles mean terrain will cause fewer shadows, but nightfall becomes inevitable. At first glance, it might seem that the only feasible surface exploration strategy would employ a rover with the means of surviving the extreme thermal range of lunar day and night. Closer analysis shows this is not the case.

5.2 Navigation Strategy

An alternate strategy avoids nightfall by circumnavigating the pole. By travelling in a direction opposite lunar rotation, synchronized with the sun, a vehicle could maintain day conditions as in Figure 5-2b for months at a time. The selected latitude of travel must balance at least three competing pressures: to maintain higher solar elevation angles to minimize shadowing for solar energy and to keep the rover warm; to prevent the rover from overheating; and to reduce the circumpolar distance to enable the traverse at reasonable speeds. Surprisingly, the length of the lunar day and small diameter the Moon result in a required average speed of 0.37 m/s at a constant latitude of 5°[1]. The high available solar energy, unattenuated by an atmosphere, provides ample power for locomotion and other activities. The combination of long day, low radius and high solar energy is even better on Mercury [77].

The advantages sun-synchronous navigation are significant. Staying in sunlight means a rover can rely entirely on solar energy - the technical, economic and political challenges of using radioisotope power sources can be prohibitive. Maintaining a consistent solar geometry also simplifies the vehicle design. Solar arrays can be pointed in a fixed direction on the rover, removing the need for complex and heavy gimbal mechanisms. A vehicle can be designed for a narrower range of operating temperatures, and thermal radiators can be pointed in a direction opposite the sun to improve radiation efficiency. Consistent light simplifies navigation using optical cameras. Of course, circumnavigating a polar region of a planet or moon presents other enormous challenges - among them, vehicle endurance to enable hundreds or thousands of kilometers of travel, and reliable rover autonomy software to sustain travel during periods out of view of Earth (for the Moon), or too distant to enable real-time control from Earth (Mercury).

However, a scaled-down version of sun-synchronous navigation improves solar-powered polar exploration over a regional, rather than global scale. Poleward of the arctic circles of Earth and Mars, a rover could follow path circuits in an area of scientific interest, and synchronize its travel with the sun. During summer months, falling prey to nightfall is no longer a danger, but the problems of power management and thermal regulation remain. Without the requirement to circumnavigate the pole, the vehicle endurance issues diminish substantially. Sun-synchronous navigation over a region simplifies solar geometry, enabling a simpler rover design, and provides access to areas that can be circumnavigated with cyclic paths.

A key challenge in enabling access to regions of terrain is to ensure that paths are repeatable. Under solar power with re-chargeable batteries, a sun-synchronous route must charge the batteries to at least the minimum required level to execute the next day's route. Ideally, a rover would follow a circular route, and point its solar array continually toward the sun (see Figure 5-3). In the diagram, the array points radially outward to avoid shadows cast by centrally-located features. As the Earth rotates, the vehicle orientation rotates a corresponding amount by following the path

1. Of course this speed ignores the increase in path length required to avoid the massive terrain obstacles known to be prevalent near the lunar poles.

arc. As long as the charge rate from solar power equals or exceeds the power consumed through driving and other operations, the batteries will remain sufficiently charged.

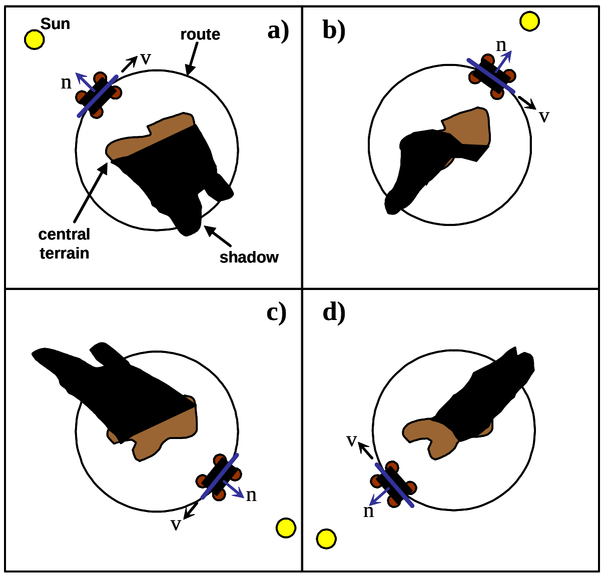


Figure 5-3: Idealized Sun-Synchronous Navigation

This idealized model is flawed in several ways. First, it is undesirable to force the vehicle to remain in motion - scientific measurements often entail extended contact with rocks or soil, or require a very stable platform. Second, vehicle power loads may outweigh available solar power, forcing periodic periods of stationary battery charging, with corresponding reduction in circuit path length. Third, medium and large-scale terrain will significantly divert a rover from a circular path.

These added complications make planning sun-synchronous routes a difficult task for humans. The objective of an automated sun-synchronous navigation planner is to take these complications into account, and still produce plans that can be repeated over much of a summer season.

5.3 Field Experiment

The Sun-Synchronous Navigation project explored a range of issues surrounding the regional variant of the sun-synchronous strategy described above, from robot mechanical and power system design to automated planning and execution software. Of greatest importance, the project sought to perform tests with a real robot executing sun-synchronous paths in a planetary-relevant polar environment on Earth.

Automated path planning was a central research topic of the project. Research into path planning for planetary exploration had to date focused on local navigation. Sun-synchronous navigation was by definition concerned with global issues as well as issues distinctly outside the realm of obstacle avoidance - terrain features that cast shadows and interrupt travel, the temporal effects of planetary rotation and maintaining sun synchrony, and energy management.

The field expedition conducted component and system-level tests on Devon Island in July of 2001. It culminated in two 24-hour, multi-kilometer long experiments designed to prove the concept of sun-synchronous navigation. In both experiments, the objective was to operate over long distances under solar power, with minimal human intervention, and to return to the start position 24 hours later with equal or higher battery state-of-charge than at the beginning. As the ambition was to prove a navigation strategy, no science or exploration activities were pursued by the rover.

The results for both experiments are described later in this chapter. Before detailing the approach for planning or the experimental results, the next sections briefly describe the operational environment on Devon Island, the Hyperion robot, and the software architecture used in the tests.

5.3.1 Devon Island

Devon Island is extremely well-suited to testing sun-synchrony. The test site was at approximately 90° W longitude and 75° North latitude - above the Arctic Circle and hence appropriate for regional sun-synchrony. In the summer months, much of the terrain is uncovered by snow or ice, exposing terrain texture that is essential for local navigation using stereo vision. Also, it is largely devoid of plant life that would also otherwise complicate automated obstacle detection algorithms not designed for seeing brush, grass or trees. Though much of the terrain is smooth, there are significant small and large scale terrain obstacles to prohibit the ideal, circular sun-synchronous path. The Arctic sun provides about 850 W/m^2 of power, sufficient power for locomotion, and yet little enough to provide a challenge for solar-powered rover designers.

5.3.2 Hyperion Rover

Hyperion was designed and built under the Sun-Synchronous Navigation project. The rover is approximately 2.4 meters long and 2.0 meters wide [78], on the scale of the rover proposed for the Mars Science Laboratory mission slated for 2009. In its configuration in the Arctic, its mass was 156 kg, 18 kg lighter than the NASA Mars Exploration Rovers[1]. It is solar-powered, and employs a 3.5 m^2 solar array of roughly 10% overall efficiency, and lead acid batteries that provide roughly two hours of locomotion power without recharging. For sensing, Hyperion made use of a stereo camera pair mounted on the front mast, and made minimal use of a scanning laser designed to provide an

1. Hyperion carried no science instruments, little communications electronics, no thermal control and no mechanical hardware to enable it to stow into a small volume and deploy for operations.

extra layer of security against obstacle collisions. For this project, stereo camera-based local obstacle detection software enabled the rover to drive at a maximum speed of 30 cm/s, or 1080 m/hr.

Hyperion's solar array is fixed in orientation, pointing to the left side at an elevation angle of 22° to optimize solar incidence over the day. Fixing the solar array orientation greatly simplified the rover mechanical design. Enabling a large solar panel to rotate but keeping it stiff under driving and wind-induced loads requires a very substantial gimbal mechanism. Also, the rover design using a gimballed array must keep the panel's swept volume clear of protruding components. Interestingly, though, the fixed solar array places a much heavier burden on automated planning - energy collection becomes highly coupled to driving direction.

5.3.3 Software Architecture

One of the objectives of the rover design was to develop software that would enable a high degree of autonomy. Given the rapid pace of development for Hyperion, it was recognized early that the system was not likely to be fully autonomous in the first year. It was also understood that low and high-level testing benefits from an architecture that permits an operator to select the degree of control autonomy based on the task at hand. Hyperion's software architecture was designed to provide "sliding autonomy", from manual control of individual motors on one end of a spectrum, to fully autonomous operations on the other.

This chapter is mostly concerned with high level autonomy, in relation to TEMPEST and autonomous navigation in general. Autonomy was somewhat limited in Hyperion's first field season. Figure 5-4 shows a diagram of the software modules relevant to this first year's operations

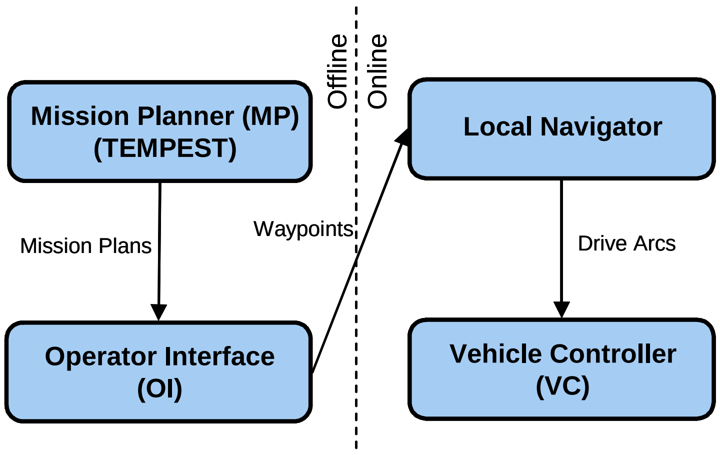


Figure 5-4: Hyperion Autonomy Software Modules

TEMPEST was Hyperion's off-board Mission Planner (MP). This early version of TEMPEST was substantially slower than the current version, and did not yet incorporate ISE re-planning. Because it could not yet respond to state or model updates during execution, there was no reason to integrate it into the online software. TEMPEST was run off-line to produce plans that were used for an entire 24-hour experiment. Despite the off-line distinction, TEMPEST provided an autonomous capability to the human operators.

An Operator Interface (OI) utilized a graphical user interface and a direct communications link to send commands and receive telemetry from the robot. Noticeably absent from the software architecture is an executive process to control and monitor the progress of plans. Human operators assumed this role via the OI. Plan Drive target waypoints were sent individually and manually to the rover at the times specified in the plan.

Between periodic manual interventions to send plan actions, Hyperion performed local navigation autonomously. The Local Navigator module used stereo vision to detect obstacles in its path, and called upon the D* algorithm to find optimal paths to the waypoints designated in the TEMPEST plans [71]. Local Navigator goals were goal regions rather than points. TEMPEST, as a global planner operating on coarse maps, cannot precisely designate goals. Therefore, it is inappropriate to treat goals as precise when passed to the Local Navigator. Furthermore, point goals are likely to be placed in the midst of intraversible terrain, preventing the rover from achieving them. Local Navigator goal regions were rectangular regions whose long axis was perpendicular to vector between the rover and the goal point, 30 meters wide and 10 meters deep. In practice this led to a behavior in goal seeking that maintained better solar array sun pointing if local obstacles diverted the rover from its original course.

5.3.4 Planning Problem

The main objective of system-level experiments was to demonstrate 24-hour solar-powered operations over as large a circuit as possible, and to complete the circuit with the same or more battery charge than the at the start. Rather than fully selecting the route and distance for the sun-synchronous circuits, TEMPEST addressed a sub-problem. Visual inspection of the terrain on Devon Island revealed that there were many terrain hazards smaller in scale than could be represented by the elevation map (25 meters spatial resolution), particularly steep-sided riverbeds. Furthermore, Hyperion's local navigation system was not sufficiently capable of autonomously detecting and avoiding these same unrepresented hazards. Wisely, the team elected to perform the system-level experiments by pre-surveying a benign sequence of via points, using a handheld GPS. Adjacent points were separated by varying distances, but typically by hundreds of meters.

TEMPEST was responsible for selecting the specific route between the sequential via points, separated by roughly 250 meters on average. *More interestingly, because the route was partially selected by humans, the planning problem became principally one of selecting the route timing to optimize battery energy management.*

5.4 Planning Approach

Sun-synchronous planning was performed using TEMPEST, albeit a very early version. Because of the early stage of development, many planning details were different than described in Chapter 4. Despite the differences, the terminology and notions presented in Chapter 4 are sufficient to describe a majority of the approach. Table 5-1 summarizes the TEMPEST parameters used for the field experiment.

Planning closely mimicked the sequential goal planning problem described in Chapter 4, with each pre-designated via point acting as a Via Point goal (see Chapter 4, Section 4.2.4). TEMPEST used a four-dimensional search space under the ISE BESTDPARMS mode (see Chapter 3) to find the path requiring the least initial energy, subject to an upper bound on mission duration. The objective function, used to compute path costs relative to the upper bound, minimized total plan duration. Given the problem of sun-synchrony, TEMPEST constrained plans to be under 24 hours. Under that upper bound, TEMPEST found the plan with the lowest battery energy. Finally, because the underlying objective was to achieve lowest energy paths, the ISE dominance mechanism (see section 3.2.3) was used to remove energy-dominated states from consideration.

However, TEMPEST deviated significantly from the algorithm and methods described Chapter 4. As mentioned earlier, a principal objective of planning for sun-synchrony is to determine the optimal start time. This conflicts with the method in Chapter 4 in which the start time is presumed to be known. To determine the optimal start time, a user designated a 24-hour time interval representing the allowable range of start times. Segment initialization followed the general approach listed in the INIT_SEGMENTS(S, Γ, k) function of Chapter 4, but with an important distinction. Rather than projecting the latest allowable time to the final goal, then using the fastest possible speed to set the latest allowable times for earlier segments, TEMPEST imposed the slowest allowable time on all goals in the sequence. Referring to Figure 4-8, rather than imposing time interval bounds as shown in frame b), this early version of TEMPEST imposed the limits shown by the dashed lines in frame a), a much more conservative approach. However, coupled with the 24-hour start time interval, TEMPEST was still very free to pursue a wide range of time profiles.

Table 5-1: Sun-Synchronous Navigation Planning Parameters

Feature	Description
World Model	Terrain: elevation, slope Ephemeris: CSPICE Solar Flux: constant value during daylight
Rover Model	Locomotion: simple force model (friction, gravity) Power: Solar array, re-chargeable battery

Table 5-1: Sun-Synchronous Navigation Planning Parameters

Feature	Description
State Space	IPARMS: • x, y (position) cells; resolution: 25 m DPARMS: • t (absolute time) sec; CSPICE ephemeris time; resolution: 1 sec / 20 min • e (battery energy) Joules; resolution: 1 J / 20,000 J
Action Set	Mobile: • Drive Actions: one action for each of eight adjacent map cell neighbors Stationary: • Charge Action: θ : solar optimal; Δt : 20 minutes
Constraint Set	Max. slope, max. battery charge
Mission Specification Set	24-hour start time interval Start position, energy Sequential Via Point goals (no goal actions) Final goal energy
Planning Details	ISE Mode: BESTDPARMS Objective function: Δt Better: if $class(t_1) = class(t_2)$ and $e_1 < e_2$ Dominates: if $class(t_1) = class(t_2)$ and $e_1 < e_2$ Re-Planning: none Special: • Path selection heuristics • Latest allowable time limit imposed on every goal

In planning, rather than using a single query for the first plan segment, TEMPEST performed start queries over the entire start interval as it would for an intermediate segment. Rather than use the objective function to distinguish between plan solution over the start window, TEMPEST used a series of heuristics to find the best path. We define the set of candidate plans derived from repeated start queries over the 24-hour start time interval to be:

$$\Pi_{cand} = \{\pi_1, \ldots, \pi_n\} \qquad 5\text{-}1$$

where each plan is defined by waypoints at listed in Chapter 4, Equations 4-1 and 4-2. TEMPEST applied three heuristics to select the optimal plan from the list of candidates:

<u>Get Minimum Initial Energy Plans:</u> Given the definition of energy e_i at each waypoint, a low initial energy corresponds to the least initial battery charge required to achieve the goal state:

$$\Pi^*_{ie} = \mathrm{argmin}_{\pi}(e_0) \qquad 5\text{-}2$$

Get Minimum Peak Energy Plans: The peak energy in a plan represents the greatest demand on the battery over the traverse. Finding the minimum peak energy minimizes the peak demand on battery reserves:

$$\Pi^*_{pe} = \mathrm{argmin}_{\pi}(\max(e_i)) \qquad 5\text{-}3$$

Get Earliest Start Time Plan: Given a list of available start times, in many cases it is operationally sound to select the first opportunity path, leaving fallback options in the event of failure:

$$\Pi^*_{st} = \mathrm{argmin}_{\pi}(t_0) \qquad 5\text{-}4$$

Note that not all start times may be feasible, and that there may be one or more ranges of feasible start times. These are determined automatically by TEMPEST. The heuristics were applied sequentially as a composite selection criterion to produce the optimal solution:

$$\Pi^*_{opt} = \Pi^*_{st}(\Pi^*_{pe}(\Pi^*_{ie}(\Pi_{cand}))) \qquad 5\text{-}5$$

In words, the composite criterion first selects on the basis of minimum initial energy requirement. From those paths, it selects on minimum peak requirement on energy over the path. From those plans remaining, it selects the one that departs the earliest.

5.5 Experiment 1 Results

Table 5-2 summarizes the results of Experiment 1 mission planning and execution. Experiment 1 was the first system level sun-synchronous navigation trial of the project. The execution of the planned route was highly successful, proving the feasibility of the strategy. The following subsections analyze the TEMPEST plan generated for the experiment, and provide the results from its successful execution.

5.5.1 Planning

Though the rover was capable of driving at 1080 m/hour, the first experiment was designed with a conservative target distance. The planned total distance was 5.6 km, to be covered in just under 24 hours, for an average speed of 236 m/hour, or 22% of the maximum.

Table 5-2: Summary of Experiment 1 Plan and Execution

Quantity	Planned	Executed
# of Goals	19 (last goal was also the start position)	
Goal Spacing mean/min/max (m)	269 / 125 / 481	
# of Actions	196 (183 Drive/13 Charge)	
Representation Factor (f_R)	1.0444	-
Avoidance Factor (f_A)	1.0479	-
Loiter Factor (f_L)	1.2212	-
Distance (m)	5600	5980
Duration (hh:mm)	23:45	24:01
% within 10 deg of optimal pointing	32.5	31.5

The field team pre-designated 19 Via Point goals to guide Hyperion away from hazardous mid-scale terrain hazards. The separation between goals was on average 269 meters. Figure 5-5 shows the Via Point goals and the selected route superimposed on a contour map of the terrain. The route progresses clockwise through the Via Points, starting just above the bottom right extremity of the circuit.

The Via Point goals were selected to guide the robot away from hazards that might be difficult for the robot to detect. The positioning of the Via Points indicates how hazardous terrain prevented an ideal circular path. Streambeds ran along the outside of both diagonal legs of the route, and a rocky promontory rose just West of the northwest end of the route. The pre-designated Via Points steered clear of these terrain hazards, but forced an elongated shape for the path circuit - a major deviation from the ideal circle. Long, straight paths prevent a fixed-orientation solar array from tracking the sun. The challenge for TEMPEST was to select the route and timing to maintain battery energy despite inevitable solar array mispointing.

The rover start and final goal position is located at the point labeled Return To Start at the southeast (bottom right) of the plot. The plan follows the circuit in a clockwise direction to match the motion of the sun in the sky, for a total planned distance of 5.6 km.

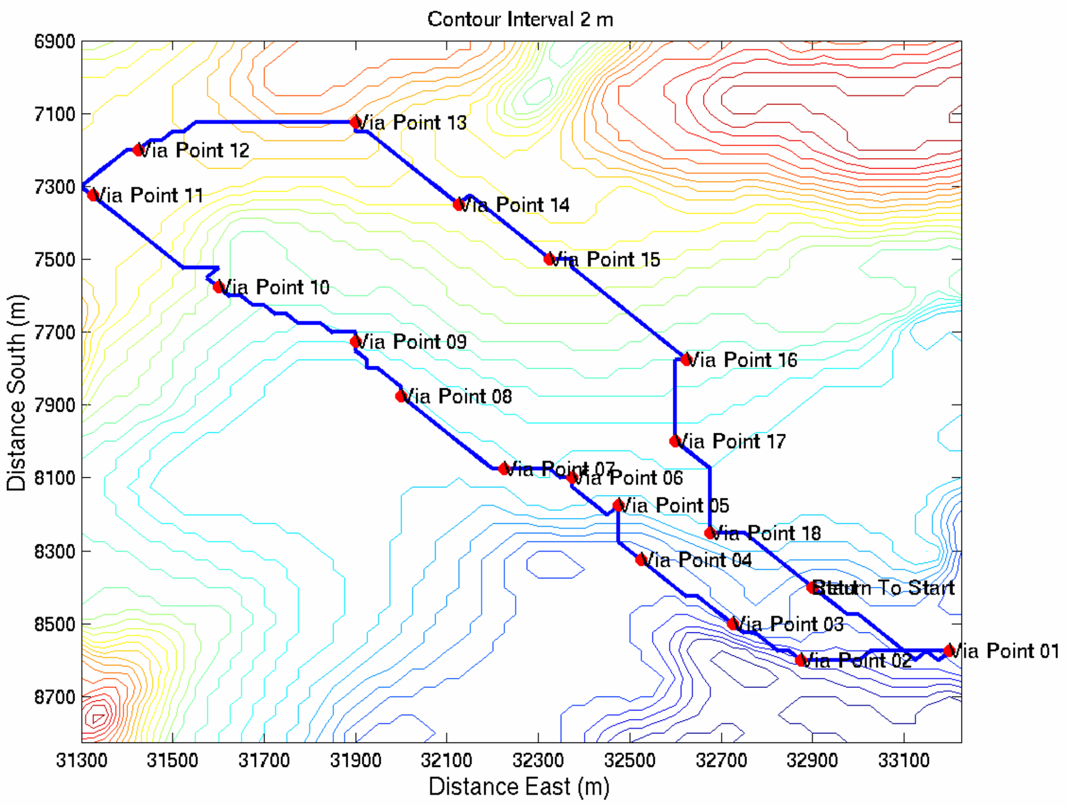


Figure 5-5: Experiment 1 Route and Elevation Map

Figure 5-6 plots Experiment 1 results versus Universal Coordinated Time (UTC). Given the longitude of the test site, the local time is computed by subtracting 6 hours from the UTC time. Figure 5-6a) is a plot of progress distance, mimicking the sequential planning diagrams in Figure 4-8. The dashed line to the left of the figure is the line of fastest approach, whose slope is $v_{max} = D_{map}/\Delta T_{min}$ (see Section 4.5 for definitions). It emerges from the opening of the start time interval, and denotes the earliest reachable portion of the state space. Meanwhile, the dashed line to the right of the figure is the line of minimum allowable rover speed. Since the goals for these experiments were Via Point actions without duration, the slope of this line represents the lower bound on minimum speed $v_{min} = v_{max}/(\tilde{f}_R\tilde{f}_A\tilde{f}_L)$. For both sun-synchronous experiments, the worst-case time increase factor $(\tilde{f}_R\tilde{f}_A\tilde{f}_L)$ was set to 1.265.

The labeled points indicate the planned arrival time at each goal Via Point, and their cumulative radial distance ρ from the start position, through all earlier goals. The solid trace through the goals is the progress distance of the plan versus time. One immediately sees that the average speed of the plan, as denoted by the average slope, is slower than both the fastest possible approach line and the slowest allowable approach line. By multiplying the duration factors listed in Table 5-2, one computes the overall duration factor to be 1.337, larger than the factor imposed on the slowest allowable approach line. One also observes that the major component of this increase is the loiter factor. The Experiment 1 plan included 13 Charge actions. At 20 minutes each, stationary activities account for over 4 hours of the plan.

Figure 5-6b) displays the Experiment 1 minimum required battery energy predicted by TEMPEST, on the same time scale. The target end-of-traverse battery charge was set at 194 W-hr (700 kJ). Observe that, with exception to the final ascent to the goal battery state-of-charge, the plan maintains the required battery energy well below the maximum battery capacity of 250 W-hr, shown by the upper limit of the error bars. The peaks in the energy curve correspond to conditions following Charge actions, and anticipate the times of greater energy demand in the plan. The final ascent of the curve at the end of the plan meets the demand placed at the final Via Point goal - to achieve an energy of 194 W-hr. The prominence of the final goal energy requirement relative to the rest of the plan indicates that it was set too conservatively to enable the route to be repeated on successive days. In theory, the final goal energy could have been set to the required energy at the start of the plan.

What explains the peaks and troughs of energy demand earlier in the plan? To answer this question, we examine the time-varying lighting and the solar array sun angles imposed by the plan. The five frames of Figure 5-7 depict snapshots of the planned route, the projected lighting on the terrain, and the sun and solar array orientations at various times for Experiment 1. The varying shading of the background reflects the changing average sun angle of incidence on the terrain, from just before local noon in Figure 5-7a to roughly one hour before local midnight in Figure 5-7d. Predicted shadows appear as black regions in Figure 5-7d. Vectors representing the sun direction and solar array normal emanate from the rover position in each frame. Note that the solar array normal points 90 degrees to the left of the driving direction, as it does for Hyperion (see Figure 5-1).

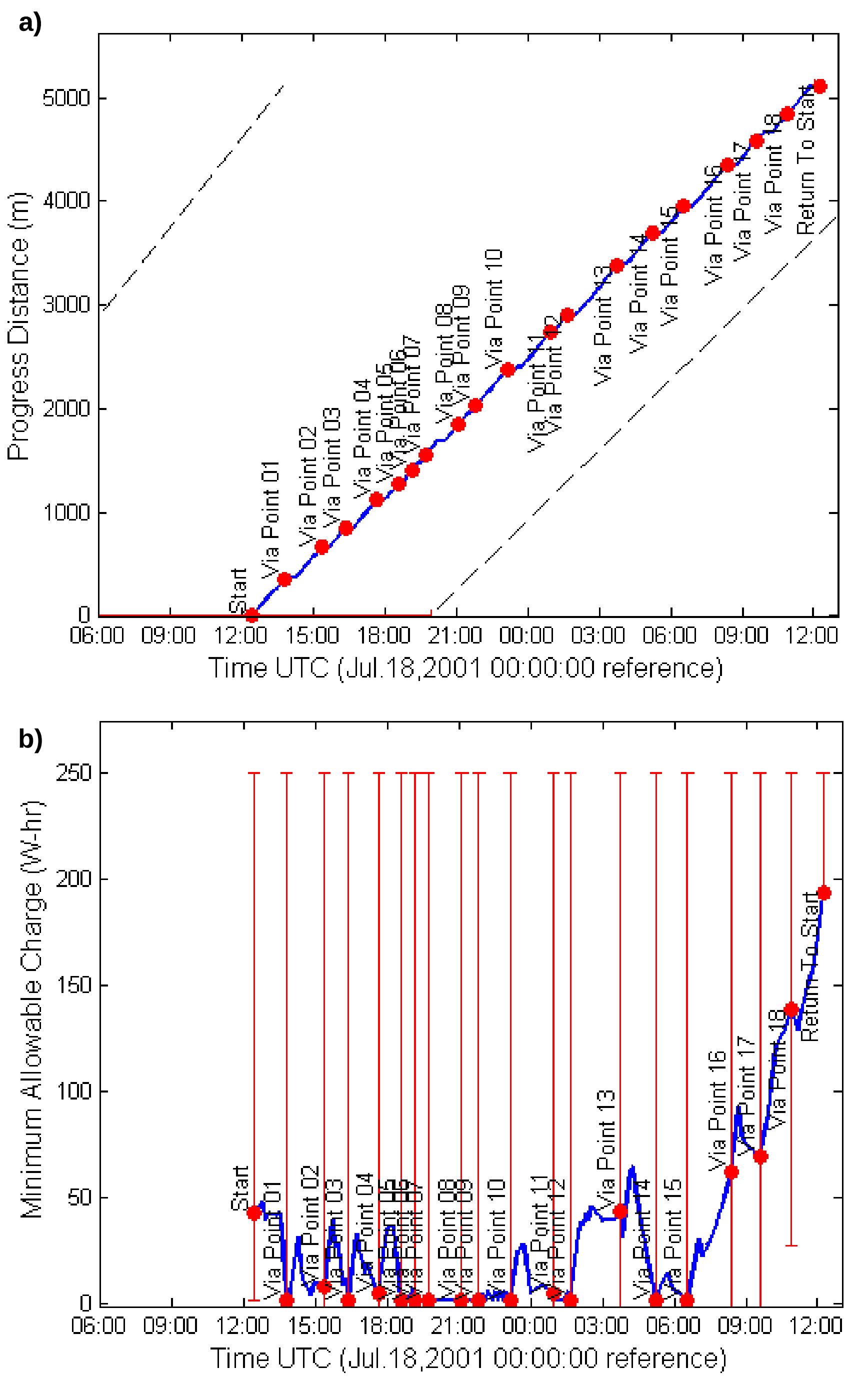


Figure 5-6: Experiment 1 Results: a) Progress Distance; b) Required Minimum Battery Energy

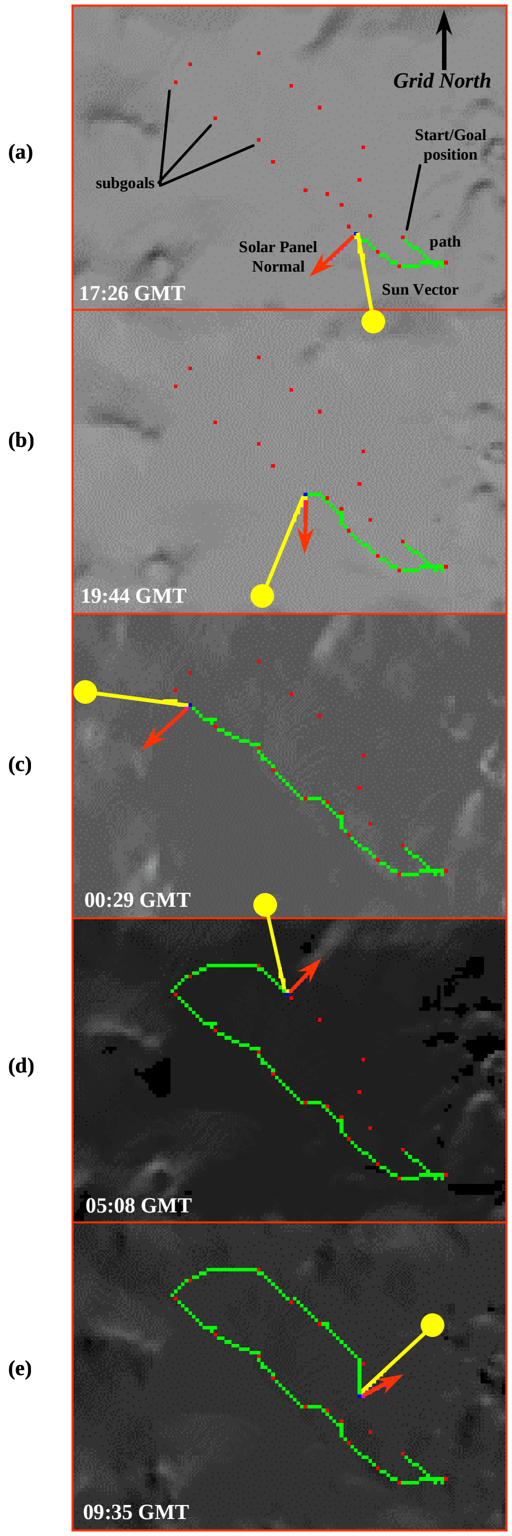


Figure 5-7: The Experiment 1 plan shows how TEMPEST achieved an optimal energy path under terrain and solar array constraints

Figure 5-7a shows an early snapshot of the path. Note how the sun direction is aft of the solar array normal. Meanwhile, in Figure 5-7c, the sun direction is forward of the solar array normal. These snapshots reflect TEMPEST's ability to schedule the path to achieve the best average sun angle, considering the entire path rather than favoring any path segment. The length of the northwest leg in these frames, coupled with a limitation in rover top speed, prevented the plan from achieving an ideal sun angle at every step. In an optimal compromise, TEMPEST selects timing that achieves the best *average* sun angle. It biases the sun *aft* at the beginning of the leg (Figure 5-7a), in anticipation that the sun will overtake the rover near the center of the leg (Figure 5-7b), and biases the sun *ahead* of the rover by roughly the sam angle at the end (Figure 5-7c). A similar behavior occurs in Figure 5-7d and Figure 5-7e on the return, southeast leg of the traverse.

Referring back to Figure 5-6, observe that the least energy-demand occurs between Via Points 5 and 10, or over the time range from 19:36 to 23:09 UTC. This matches the time period where the sun was closest to normal to the solar array. Furthermore, observe the demanding range of the plan between Via Points 12 and 14, from 1:40 to 5:14 UTC. This time range corresponds to northernmost extent of the route, and some of the largest aft sun angles, biased to average out the return leg of the cycle. Furthermore, the time is approaching local midnight, the time of least incoming solar energy of the day.

A measure of TEMPEST's ability to maintain sun exposure on Hyperion's solar array is shown in the histogram in Figure 5-8a. The histogram depicts the relative azimuth angle, in the plane of local horizontal, from the sun to the solar array normal resulting from the *plan* for Experiment 1. Zero degrees relative azimuth indicates optimal pointing, while negative and positive values indicate sun-aft and sun-forward conditions, respectively. The histogram indicates that over 32% of the route was spent within 10 degrees of optimal solar array pointing. We attribute the bias in sun angles toward the aft of the rover to the Get Earliest Start Time Path heuristic (see Section 5.4). Following the application of energy-based heuristics, TEMPEST selects the earliest path opportunity. For clockwise paths in the northern hemisphere, earlier opportunities will bias sun angles aft, as the Earth has not progressed as far in its daily rotation.

The second histogram, in Figure 5-8b, depicts the same quantities for the *executed* Experiment 1. Qualitatively, one can immediately see the similarities between the profiles, indicating integrity of the execution to the mission plan. Given how closely the plan was followed during execution, differences in these profiles are likely due to off-pointing that occurred during actions taken by the Local Navigator to avoid obstacles [71].

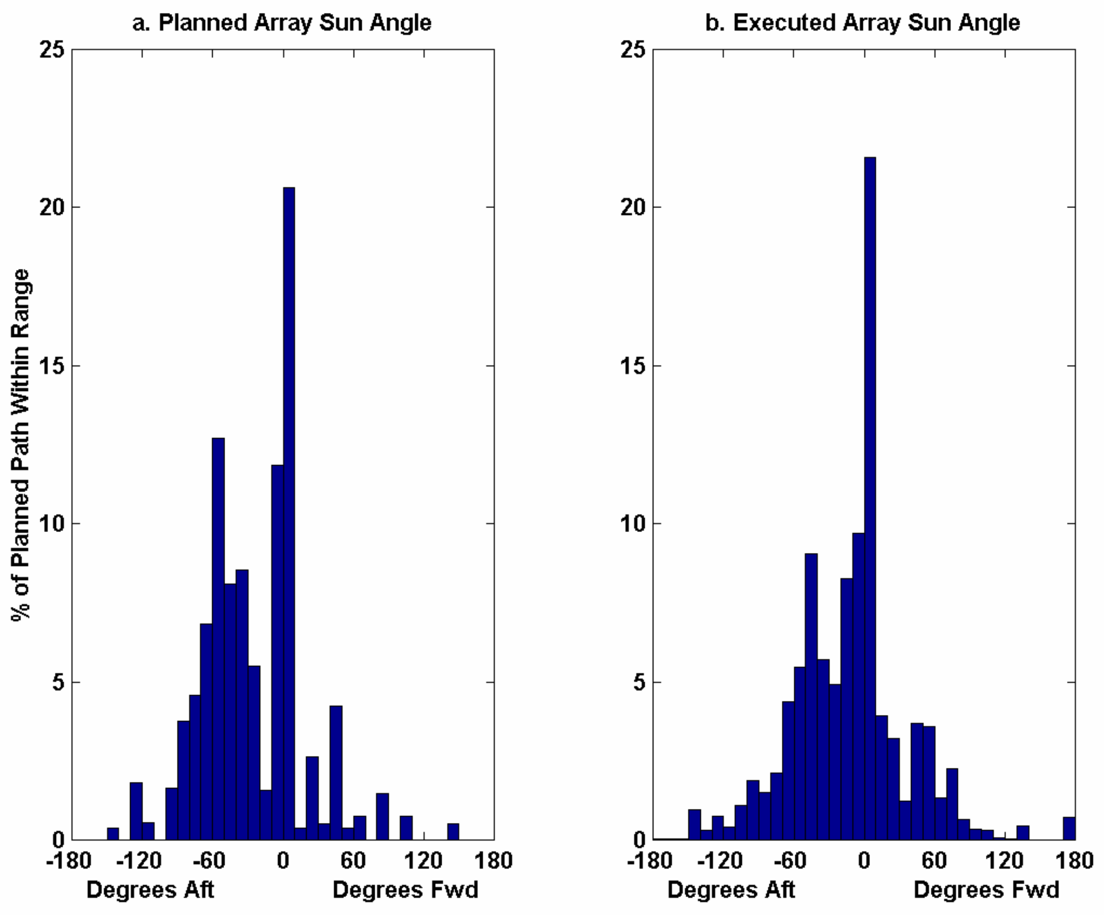


Figure 5-8: Experiment 1 Planned and Executed Solar Array Sun Angles

5.5.2 Execution

The Experiment 1 mission execution proceeded smoothly over entire the 24 hour test. Because the nominal rover speed given to TEMPEST for planning was so conservative (8 cm/s as compared to 30 cm/s), Hyperion typically completed each Drive action well in advance anticipated by the plan. However, to keep on schedule, human controllers refrained from sending the next Drive action target until the start time specified in the plan. Therefore, the motion of the rover was uniform, but rather alternated between its top speed of 30 cm/s and rest at each of the Drive action goals. Figure 5-9 shows the Experiment 1 executed progress distance as a function of time, compared to the plan. With exception for a few delays, the executed rate of progress matched the plan very closely.

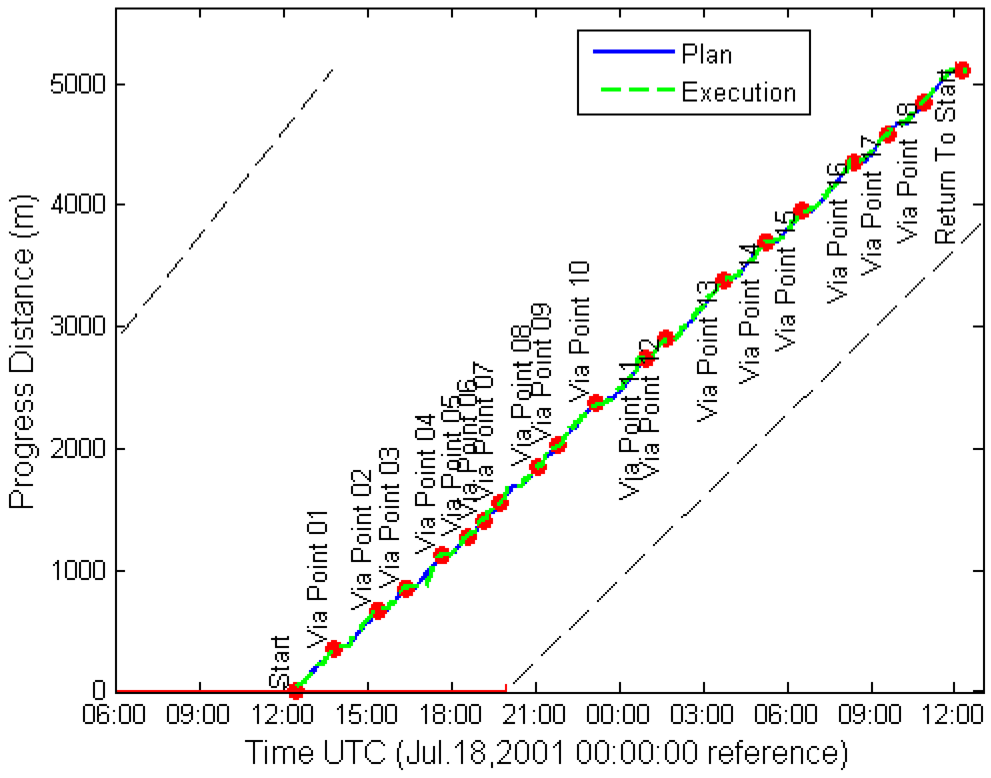


Figure 5-9: Experiment 1 Executed Progress Distance: Execution followed the plan very closely

Perhaps the best measure of success is in how successfully the plan enabled battery charge management. Due to high noise levels in current measurements recorded on Hyperion during the experiments, the battery charge could not be estimated. However, battery voltage provides an indirect measure of whether the battery is charged or is approaching discharge. Battery voltage should remain roughly constant over a wide range of charge states, but will begin to drop as the battery becomes low on charge. The battery voltages plotted in Figure 5-10 show that the plan maintained the battery potential near the nominal 24 Volts for the entire 24-hour mission.

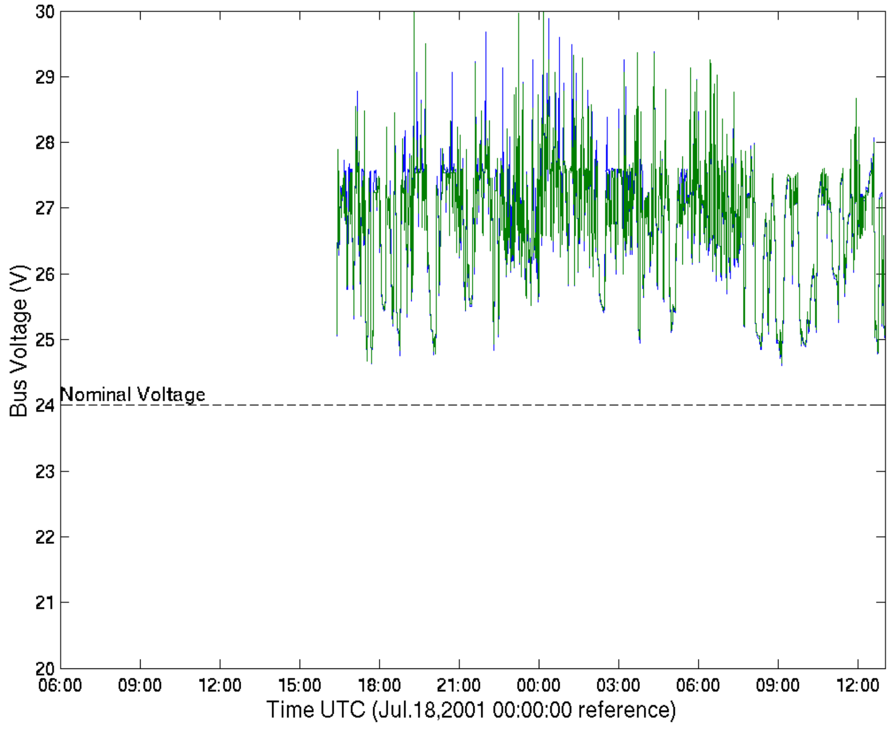


Figure 5-10: Experiment 1 Battery Voltage: Data indicates that the batteries did not show signs of extreme discharge.

Though Experiment 1 proved the feasibility of the sun-synchronous navigation strategy, it did not prove whether the strategy provided any substantial benefit above more standard navigation approaches. Unintentionally, Experiment 2 provided strong evidence that sun-synchronous navigation, and TEMPEST, were essential in sustaining the rover over an extended traverse.

5.6 Experiment 2 Results

Table 5-3 summarizes the results for Experiment 2, whose planned route is superimposed on a terrain contour map in Figure 5-11. In contrast to Experiment 1, the execution of Experiment 2 was fraught with operational difficulties which caused the rover to fall far behind the schedule specified by the plan. The sharp decline in rover performance that occurred as a result of the deviation from the plan schedule clearly showed the sustaining effect of sun-synchronous navigation, and of TEMPEST planning. The next subsection analyzes the Experiment 2 plan in detail. The discussion of the Experiment 2 execution follows in a second subsection.

5.6.1 Planning

A majority of the southeast portion of the circuit for Experiment 2 followed the route of Experiment 1. However, new Via Point goals directed the route significantly farther west and north, circumnavigating a rocky outcrop informally named “Marine Peak” (see the topographic feature to the East of Via Points 9, 10 and 11 in Figure 5-11). The Via Points were placed at an average distance of 249 meters. This route, at 8.4 km, was substantially more ambitious given the success of the first experiment. To enable a plan this long, the average drive speed used in planning Experiment 2 was 11 cm/s, or 400 m/hr, only 37% of the maximum speed, but 39% faster than for Experiment 1.

Table 5-3: Summary of Experiment 2 Plan and Execution

Quantity	Planned	Executed
# of Goals	30 (last goal was also the start position)	
Goal Spacing mean/min/max (m)	249 / 71 / 559	
# of Actions	301 (289 Drive/12 Charge)	
Representation Factor (f_R)	1.0526	-
Avoidance Factor (f_A)	1.0730	-
Loiter Factor (f_L)	1.1884	-
Distance (m)	8437	9059
Duration (hh:mm)	25:05	24:09

Figure 5-12 plots Experiment 2 results versus Universal Coordinated Time (UTC). Recall that the local time is computed by subtracting 6 hours from the UTC time. Figure 5-12a) is a plot of progress distance. As in Figure 5-6a, it shows the dashed lines representing the earliest possible approach and latest allowable approach. The lines originate at the open and close of the 24-hour start time interval, respectively. Also as in Experiment 1, the worst-case time increase factor $(\tilde{f}_R\tilde{f}_A\tilde{f}_L)$ was set to 1.265. The actual duration increase factor, computed as the multiplication of f_R, f_A, and f_L as listed in Table 5-3, was 1.342, very similar to the factor from Experiment 1. However, the avoidance factor from Experiment 2 was noticeably higher, and the loiter factor, noticeably less. Though the specific reason for greater avoidance is not obvious, the reduction in loiter factor is the result of a lower number of Charge actions in Experiment 2. Four of twelve Charge actions were designated between Via Points 9 and 13, and account for the reduction in slope in the curve before the plan halfway point.

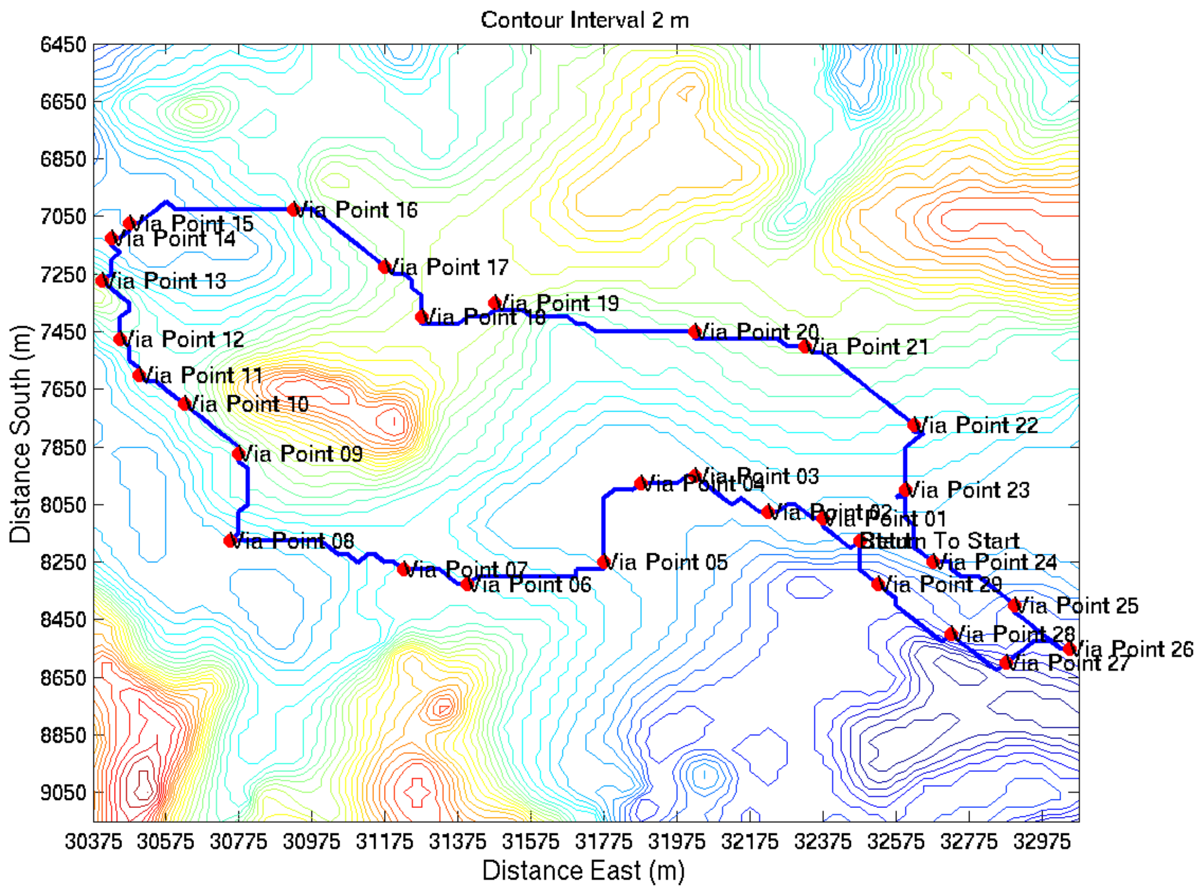
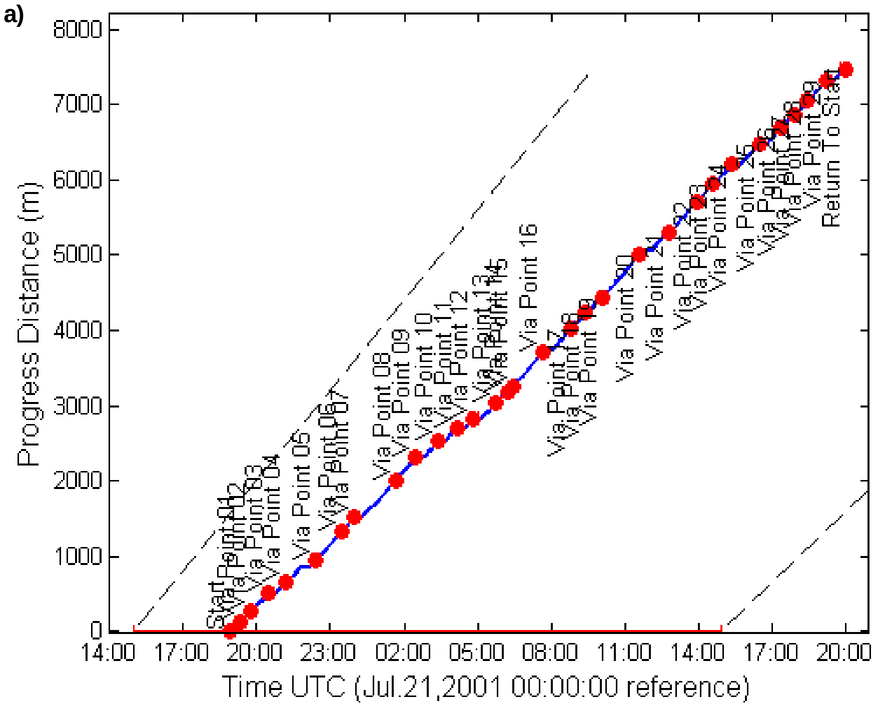


Figure 5-11: Experiment 2 Route and Elevation Map: The route starts and ends at the point marked 'Return To Start', and traversed a nominal distance of 8.4 km, including a circumnavigation around the terrain feature known informally as Marine Peak at the West end of the route.

Figure 5-12b) displays the corresponding Experiment 2 minimum required battery energy profile. The *Mission Specification* end-of-traverse battery charge was reduced to 139 W-hr (500 kJ). As before, the plan maintains the minimum energy well below the capacity of the battery, with the exception to the rise to the final goal target. Again, the target energy could have been set far lower (to empty in this case) and still would have allowed sufficient energy in the battery to repeat the route on following days.

a)



b)

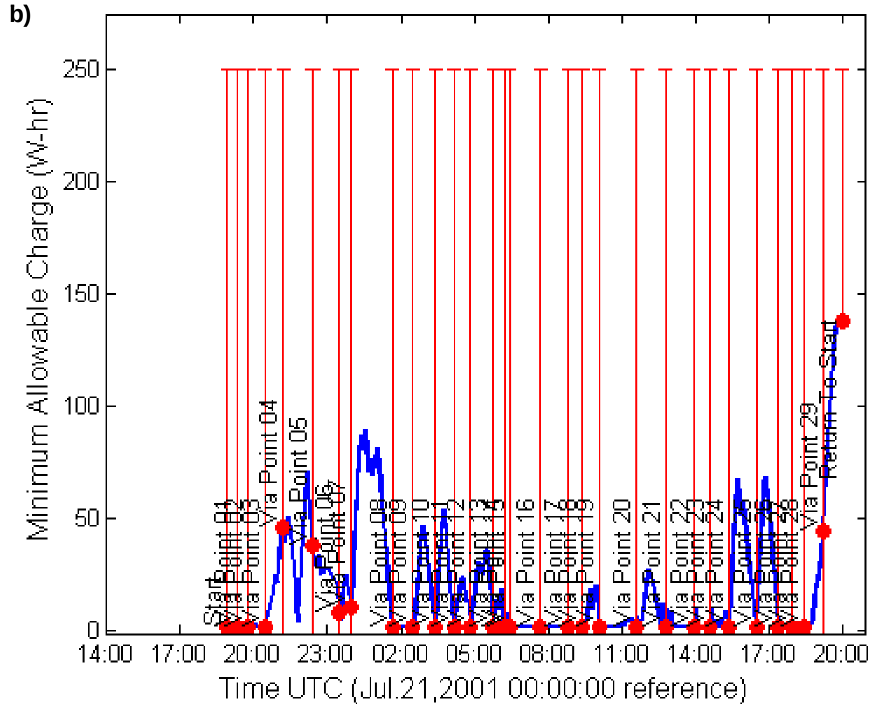


Figure 5-12: Experiment 2 Results vs. Time: a) Progress Distance; b) Minimum Required Battery Energy

Curiously, the greatest demand on battery energy, according to the Experiment 2 plan, occurs on the northwest leg of the plan (between Via Points 7 and 8, or 23:58 to 01:39 UTC) in the position where the energy profile was least demanding for Experiment 1. The least demand on battery energy appears to come between Via Points 15 and 18 (06:27 and 09:25 UTC), also curiously at the place in the route for Experiment 1 that demanded the greatest energy.

5.6.2 Execution

The execution of Experiment 2 was repeatedly interrupted near the beginning by communications loss between the operations tent sheltering the human controllers and Hyperion. Since each Drive action target was sent manually over wireless ethernet, communications outages forced the rover to stop moving. Figure 5-13 shows the executed progress distance for Experiment 2. Note that the execution started roughly an hour late. Despite the conservatism in the rover speed assumed in the TEMPEST model (11 cm/s for planning as compared to 30 cm/s maximum speed), the delays eventually caused the robot to fall way behind schedule.

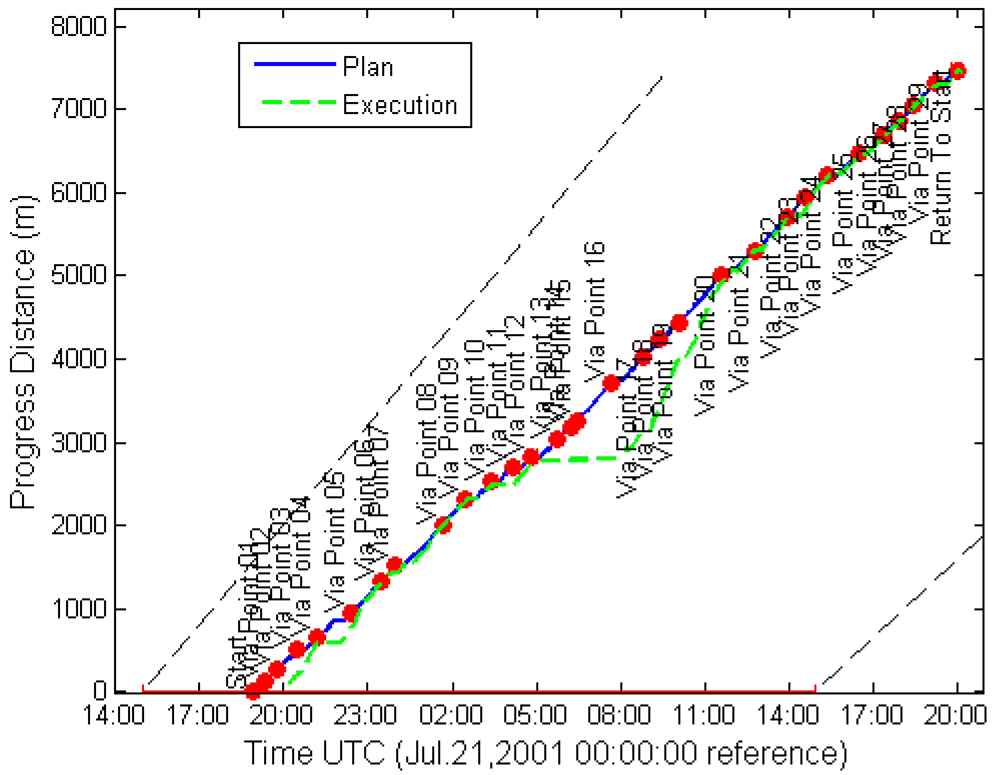


Figure 5-13: Experiment 2 Executed Progress Distance: the most substantial delay put Hyperion behind the plan by several hours, causing significant battery discharge

As time passed, the sun became more and more biased towards the front of the vehicle. During periods of good communications, Hyperion's stereo cameras became blinded by sunlight, disabling autonomous local navigation. In an effort to keep up with the plan at any cost, human operators decided to teleoperate the vehicle using panoramic imagery produced by Hyperion. Unfortunately, the panoramic imagery was also negatively affected by the sunlight, worsening the already low-resolution images. Finally, teleoperated driving caused the robot to drive into a rock that it was unable to cross. During the struggle to surmount the obstacle, Hyperion's front axle rotated beyond an angle limit imposed by software. The recovery from this final fault entailed physically moving Hyperion away from the rock and re-starting rover software. At 13 hours into the mission, Hyperion was over 3 hours behind schedule.

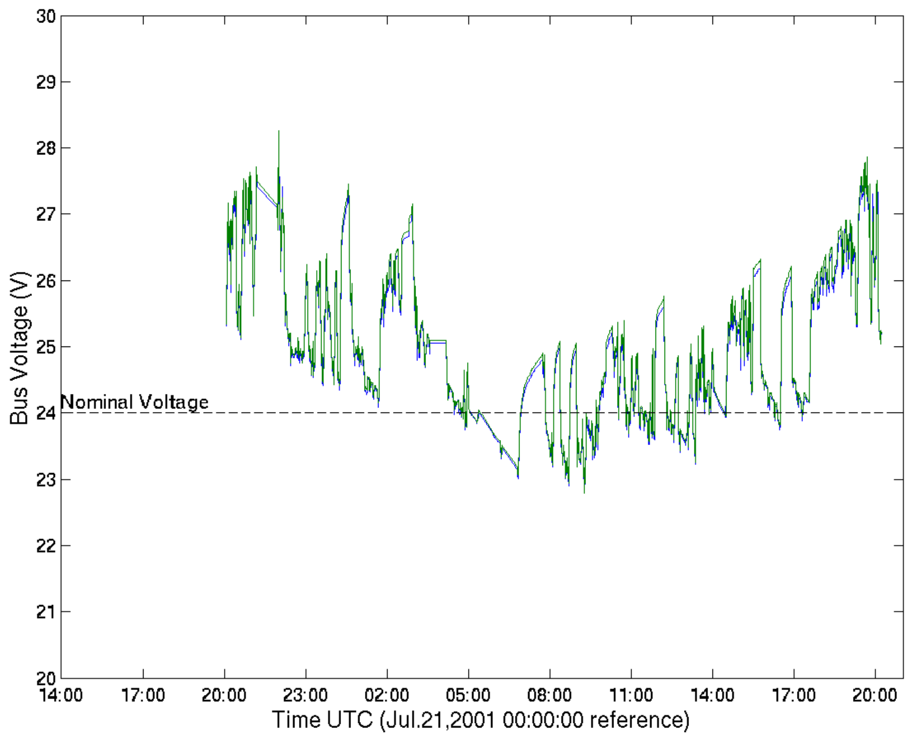


Figure 5-14: Experiment 2 Battery Voltages: evidence strongly suggests that falling behind the TEMPEST schedule, which resulted in poor sun angles, caused the substantial battery discharge during the mission.

These successive delays strongly negatively impacted rover power. Each delay caused the sun to be biased to the rover's front, or even towards the vehicle's right side (opposite the active side of the solar array). Though operating energy costs (i.e. driving, electronics, etc.) were fairly uniform over the route, the poor sun angles drastically reduced incoming energy to offset those costs. The battery state-of-charge could not be estimated due to high current measurement noise. However, the battery voltages plotted in Figure 5-14 provide a clear measure of Hyperion's declining battery charge. Unlike in Experiment 1, Experiment 2 voltages dip to below the designed bus voltage of 24 Volts, indicating a substantial battery discharge. This dip corresponds in time to the period of greatest delay.

Interestingly, once Hyperion was rescued through a manual intervention, human operators teleoperated the rover at full speed to catch up with the original plan. Human operators were repeatedly forced to command Charge actions to maintain minimum battery charge. Observe in Figure 5-13 how the progress distance trace re-acquires the plan line following the most substantial delay. Meanwhile, the battery voltage in Figure 5-14 climbs back to pre-delay levels in the latter part of the mission.

This is strong evidence that TEMPEST plans enable battery charge management, and that deviation from the plan resulted in severe battery discharge and other sub-optimal effects like camera sun blinding. It also suggests an instability in the control of sun-synchronous routes. As the rover gets further behind, the collection of solar energy is progressively more difficult to achieve, and the required speed to catch up to the plan schedule, and hence required locomotion power, increases. If the robot is required to get back on schedule before completing a full circuit, schedule delays of a certain duration will be unrecoverable. At some point, the power required for driving and survival outweighs the incoming solar power, and leads to a continual discharge of batteries. The batteries must sustain the recovery for its duration, or be fully discharged. Alternatively, if the rover is not required to catch up to the plan schedule before the completion of the circuit, the recovery speed could be adjusted to enable the system to sustain itself. Lastly, if shadowing is not expected to interfere with lighting, the vehicle could potentially stop and rotate in place to follow the sun in a survival holding pattern, and reacquire the sun-synchronous circuit on the following day.

5.7 Discussion

The sun-synchronous field experiments plainly illustrated the utility of mission-directed path planning. Sun-synchronous planning would have been difficult for humans to do by hand. Large-scale terrain obstacles prevented an idealized, circular sun-synchronous navigation path. Determining the timing to achieve the optimal sun angle balance on the irregular path would have entailed extensive trial-and-error, or a hierarchical approach that first selects the route between Via Point goals and then searches over different schedules on the route solution. Mission-directed path planning provided a number of notable benefits for rover operations:

Sophisticated Reasoning: TEMPEST reasoned about how best to time the route, despite the elongated shape of the traverse. It minimized the overall effect of inevitable solar array mispointing by balancing the lead and lag of the sun with respect to the solar array at various points in the traverse. The plan for Experiment 1 proved highly effective in maintaining battery energy during the entire traverse. Executing the plan closely maintained the batteries well above the minimum bus voltage for the entire 24 hours. More interestingly, Experiment 2 illustrated the danger of substantial deviation from the TEMPEST-derived plans. Operational delays allowed the sun to be biased well to the front of the rover, and caused the batteries to discharge below the nominal bus voltage. Re-acquiring the plan later in the mission allowed the batteries to re-charge to initial voltages.

Use of Intermediate Via Points: Intermediate Via Point goals enabled human operators to steer the route away from terrain hazards that were not represented on the low-resolution elevation model. Furthermore, the designation of intermediate goals is analogous to hierarchical approaches - prior path selection to enable time-optimal trajectory planning as in [5][17][58]. However, by selecting points rather than paths, TEMPEST was far more free to consider the coupling between route, timing and resources in selecting a plan.

Natural Integration with Local Navigation: The cooperation and mutual abstraction between TEMPEST and the Local Navigator was natural. The Local Navigator module reliably avoided rocks while moving toward TEMPEST-defined waypoints. It had no knowledge of the global path, resource usage, or timing requirements. Meanwhile, TEMPEST solved for the mission-directed plan to avoid large-scale obstacles and to manage resources, but had no knowledge of local obstacles below the resolution of the terrain map.

The experiments also highlighted a number of future challenges for TEMPEST.

Brittleness to Unanticipated Conditions: The software provided no capability for planning under uncertainty, contingency planning, or automated re-planning, and was therefore brittle to unanticipated problems. In Experiment 2, a planner capable of reasoning under uncertainty might have anticipated the possibility of schedule delays and biased the plan ahead in time to avoid crippling sun angles. Re-planning might have compensated for the delays by inserting Charge actions to replenish the battery, taking more direct routes between Via Point goals, or electing to eliminate Via Points as necessary to save time to get back on schedule.

Lack of Richness in Mission Specification: At the time of the experiments, TEMPEST was limited in how missions could be specified to it. For example, it only allowed goals to be specified in terms of position, and in the case of the final goal, minimum battery energy. Plans must often meet constraints on battery energy for other goals, or fall within allowable time ranges. Specific activities might also be assigned to goal positions, and might be governed by geometric or other constraints (e.g. goal must be achieved at least one hour before dark). TEMPEST only modeled a few operational constraints for actions, (e.g. maximum slope, minimum sun elevation angle). Other restrictions on line-of-sight to communications relays and enforcing sun-in-camera stayout zones would have been a major benefit for Hyperion.

Poor Computational Performance: This preliminary form of TEMPEST required significant computational and memory resources. Planning in a four-dimensional state space (x, y, time, energy), *Experiment 1 took 5 hours 48 minutes to plan on a laptop with a 400 MHz Pentium II with 128 MB of RAM*.

Exclusively Offline Operation: TEMPEST was run exclusively offboard Hyperion and in an offline mode for the field experiments. Slow planning and lack of re-planning capability removed the incentive to configure TEMPEST for online operation.

These lessons motivated substantial improvements in domain richness, re-planning, performance and in online operations. Follow-on field experiments on Hyperion and a new rover, Zoe, demonstrate how mission-directed path planning enable far greater navigational autonomy and improve mission planning for rover exploration.

The sun-synchronous navigation planning problem is not yet solved. It would be interesting in future work to develop planning that could specify a sun-synchronous circuit given only high-level goals like target areas for exploration and total traverse distance. It would also be intriguing to quantify the sensitivity of path cost on schedule delays, as a way of bounding the delays from which recovery is possible. Finally, this thesis did not examine the behavior of plans in more desperate situations. For example, would TEMPEST display a “tacking” strategy, as in sailing, to travel in the direction of the sun?

6. Robotic Astrobiology

Astrobiology is the branch of biology concerned with the emergence and survival of life in the universe, the effects of outer space on living organisms, and the search for extraterrestrial life. Since Earth is the only known place to harbor life, a primary activity of astrobiologists is to characterize the extreme habitats on Earth, and to identify the mechanisms used by organisms to survive under conditions that mimic the extremes of other bodies in the Solar System. A second focus is to develop enabling technologies for remote life detection so that future missions to the Solar System will be equipped to find extraterrestrial life if it exists. The Life in the Atacama (LITA) project seeks to develop technology to enable robotic astrobiology for NASA, and at the same time to conduct useful Earth science in the Atacama Desert of northern Chile [79].

In a vein of the LITA research program, the TEMPEST planner was further developed to support autonomous, wide-area scientific investigations. In the Sun-Synchronous Navigation project, TEMPEST demonstrated its capacity to select routes, coordinate route scheduling, and to manage battery energy, albeit in an off-line mode. The LITA project motivates a much higher standard for online operations, richer representations for the planning problem, and planning in support of science goals.

6.1 Life in the Atacama

The Atacama is one of the driest places on Earth, and has long been known to support very little life. The LITA project conducted two field experiments, in 2003 and 2004, involving rover field testing and biological investigation, and will culminate in 2005 in a multi-week integrated field demonstration. In that final trial, a team of scientists in the United States will direct a robotic search for life in Chile. To stress robot autonomy, scientists and engineers will have limited communications bandwidth, and will only be allowed to transmit commands and receive telemetry once per day. Their mission will be to characterize the presence and distribution of microscopic life over tens of kilometers of travel in an effort to better understand the limitations of life in the Atacama ecosystem.

6.2 Navigational Autonomy for Science

In sharp contrast with the Sun-Synchronous Navigation project, whose principal aim was navigation, the goal of the LITA project is to develop a rover and software that enable remote, autonomous scientific investigation. However, LITA's scientific approach continues to stress long-distance navigational autonomy. It has baselined a strategy to characterize regional habitats by conducting widely separated detailed surveys and faster periodic surveys along traverses between detailed survey sites. Scientists will analyze data sets from previous days and, by correlating their findings with orbital data of the region, will select new targets to test hypotheses on patterns of life. Most days will be spent traversing a minimum of hundreds of meters and often several kilometers in pursuit of new goals. Navigational autonomy must reliably transport the rover to target sites for this investigation to be meaningful.

The sun geometry for the Atacama desert favors a solar array that is horizontally mounted on a rover. This configuration eliminates the coupling between driving direction and incoming solar energy, which makes the planning problem at once easier and yet less interesting.

The strategy also depends on interleaving science observations with traverses in a single day. These phases cannot be adequately planned independently - resource availability changes throughout the day, as does sun geometry that might have a bearing on navigation. Navigation plans must incorporate the position, time and resource requirements of science activities to ensure that objectives are globally feasible.

Finally, the LITA field demonstration will not tolerate planning once per day. As was shown in the Arctic, simple delays can cause a plan to become infeasible. Rather than forcing the robot to abandon a day's activities in the event of a schedule delay or other unforeseen event, re-planning to adjust to updates in state might allow the rover to accomplish most or all of its goals.

6.3 Atacama Desert

The Atacama Desert contrasts dramatically from the Canadian Arctic of earlier TEMPEST field experiments. At 21° S latitude, continual sun was replaced by a day/night cycle closer to common experience. Solar radiation was more intense in the Atacama, however, and provided a flux of roughly 1000 W/m^2 during the field experiments in 2003 and 2004. The Atacama is far more dry - streams were completely absent, and compared to the Arctic, cloud cover was rare[1], leading to more reliable solar power for rovers. A common element between the Arctic and the Atacama was the rarity of large-scale plant life. The Atacama is a rocky, salty environment that supports life in small pockets. The result is a landscape whose appearance mimics the surface of Mars, a boon for local navigation designed for extrater-

1. However, surprisingly, humidity levels at the near-coastal sites rose dramatically at night, due to heavy fog banks originating from the Pacific, which sometimes extended well into the daylight hours.

restrial use. The Atacama presents a greater challenge to navigation that the Arctic - the terrain is more varied, and rough terrain is more common. Mid-scale terrain features, too small to be represented in maps deriving from orbital data, yet too large to be perceived by traditional local sensing, are very common.

The next two sections provide results from TEMPEST tests from the 2003 and 2004 field seasons respectively. The objectives for each season were different, and developments between the campaigns resulted in an entirely new robot and vastly updated software for the later season. Therefore, each section introduces the objectives, rover and software used in tests prior to presenting results.

6.4 Field Experiment 2003

6.4.1 Objectives

The principal objective with respect to robot autonomy was to enable fully autonomous driving of at least 1 kilometer, in preparation for integrated science experiments in coming years. Where in the Arctic, short manual interventions to send plan actions to the rover were commonplace, the new system had to operate completely without human involvement. Integrating TEMPEST with an executive and a plan monitor was essential to this performance jump, but resulted in a far more complex system. Specific to planning, the objective was to characterize TEMPEST and overall system behavior in terms of plan quality, the reasons behind re-planning, and plan stability.

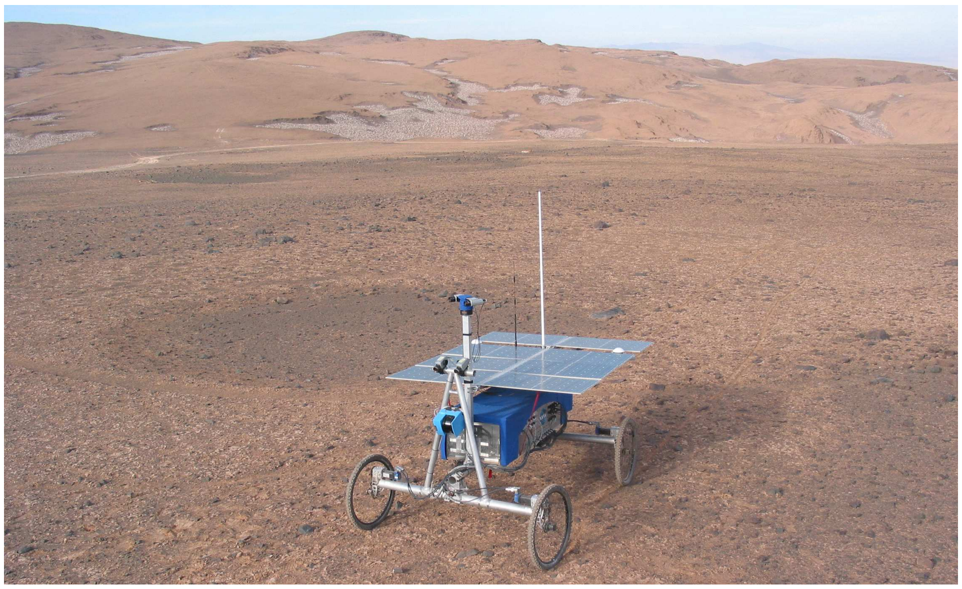


Figure 6-1: Hyperion in its LITA Configuration

6.4.2 Hyperion Rover

As described in Chapter 5, Hyperion is a solar powered robot originally designed for sun-synchronous navigation in polar latitudes [78]. Operations in a mid-latitude environment prompted a re-design of Hyperion's solar array mounting - the new configuration oriented the array horizontally to best collect energy from the overhead sun. This removed the coupling between driving direction and incoming solar power, making the energy management problem far more benign than in the Arctic. That stated, Hyperion's batteries still provided only two hours while driving without the sun's input, so it remained critically dependent on its environment. Unfortunately, Hyperion was not capable of estimating its battery state-of-charge, preventing closed-loop control under TEMPEST. At 160 kg, Hyperion's mass remained roughly the same as in the Arctic. Since it employed very similar navigation software, Hyperion's top driving speed remained at 30 cm/second, or 1080 m/hour.

6.4.3 Software Architecture

The Hyperion autonomy software comprised a Health Monitor (HM) in charge of responding to abnormal state con-

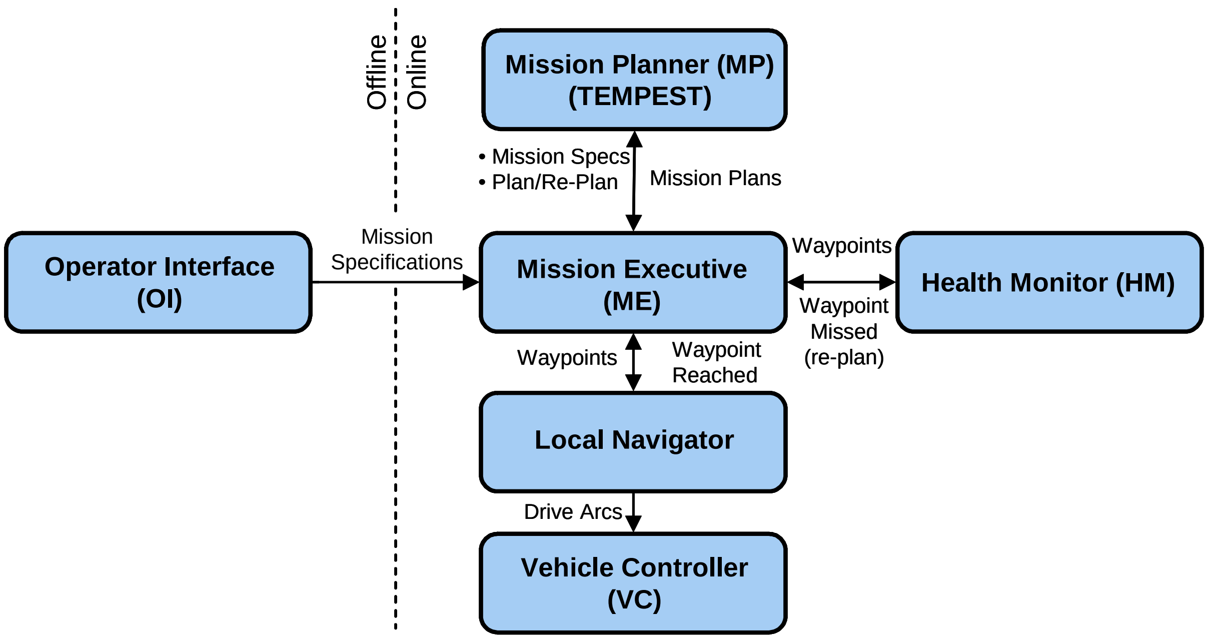


Figure 6-2: LITA 2003 Autonomy Software Architecture

ditions while operating; a Local Navigator that used stereo camera data to locate and avoid hazardous local terrain while seeking a goal; and TEMPEST which took the sole responsibility of mission planning. A rudimentary mission executive (ME) coordinated mission-related data passing between these modules, and received and distributed *Mission Specifications* from the Operator Interface (OI). Figure 6-2 illustrates the basic set of inter-module communications relating to mission-level path planning and execution. Designed as a placeholder for future, more sophisticated executive modules, the ME adopted a simple, hierarchical state machine architecture. It enacted a collection of monitors to keep track of system state while planning or executing plans: the Plan Request Monitor, Plan Execution Monitor, Drive Monitor, and Charge Monitor.

6.4.4 Sequence of Operations

The mission focus in LITA 2003 was navigation to a single distant goal. *Mission Specifications*, consisting of the goal position and required arrival battery energy level were sent via the OI. Upon receiving a *Mission Specification*, the ME determined the current robot position and time state, and requested a plan beginning at that state and terminating at the specified goal state. TEMPEST found and returned an optimal plan to the ME. For each action in the plan, the ME triggered one of two Action Monitors.

The Drive Monitor computed parameters for a 10 meter by 30 meter goal region surrounding the next position waypoint (see Figure 6-3), and sent them to the Local Navigator for execution. Using the goal region as its global goal, the Local Navigator pursued this region while avoiding obstacles it detected. Once Hyperion was within the region, the Local Navigator signaled its arrival, terminating the Action Monitor. The Charge Monitor stopped the rover and waited for the assigned Charge duration before terminating.

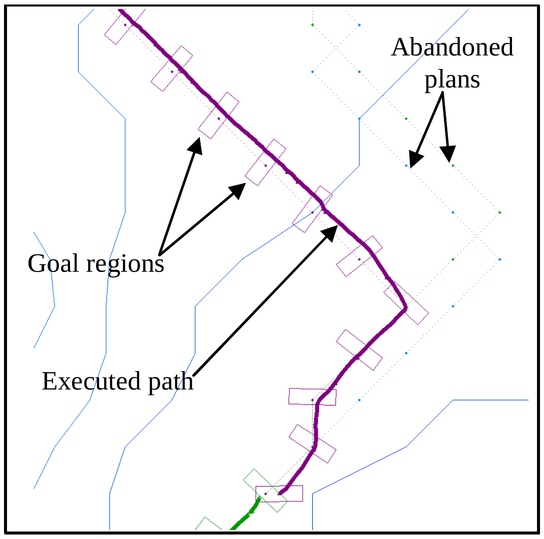


Figure 6-3: Plans and executed paths. TEMPEST plans assigned the location of periodic goal regions. Goal regions gave the Local Navigator flexibility in selecting the specific path between waypoints.

During plan execution, at the scheduled arrival time for each plan waypoint, the HM performed a one-time check to confirm the robot was on time. If the rover was more than a fixed distance from the waypoint at the scheduled arrival time, the HM notified the ME that the waypoint was “missed”. Receiving this notification, the ME terminated the execution of the current plan, and commanded TEMPEST to re-plan from the current rover state to the goal in the original *Mission Specification*.

6.4.5 Planning Approach

Table 6-1 summarizes the planning parameters used in the LITA 2003 experiments. Notably, TEMPEST was re-configured from Arctic field experiments to take advantage of the composite objective function approach (see Section 3.1.6 and Approach 2 in Section3.4.2). By representing the battery energy state variable within the objective function, the energy dimension used in Arctic experiments could be removed from the ISE DPARMS. At each state transition in the ISE search, a path would incur the corresponding positive or negative energy cost, plus the path length cost increment, defined as the absolute value of the *greatest charge* (negative cost) possible over the entire search space. The result was a dramatic improvement in planning speed that enabled TEMPEST to perform initial planning repeatedly throughout a day's experiments, if necessary. Recall, however, that in removing energy from the state space, ISE was no longer complete - solutions deleted from consideration early in the search on the basis of cost could not be resurrected if ISE failed to find a feasible solution downstream. In practice, this did not prevent TEMPEST from finding solutions.

Perhaps more important than the dimensionality reduction was incorporating ISE state update re-planning into TEMPEST. In the Arctic, TEMPEST generated a single plan for a 24-hour traverse. If operational delays prevented staying on schedule, as occurred in Experiment 2, the robot relied on human teleoperation to recover. A prime ambition for the LITA project was to obviate the need for periodic teleoperation. Re-planning in response to evolving rover state created a "safety net" in case of schedule deviations, a likely occurrence during robot experiments. Another objective from prior work was to demonstrate a richer representation of rover, actions and constraints.

Table 6-1: LITA 2003 Planning Parameters

Feature	Description
World Model	Terrain: elevation, slope Ephemeris: CSPICE Solar Flux: constant value during daylight
Rover Model	Locomotion: simple force model (friction, gravity) Power: Solar array, re-chargeable battery Camera?
State Space	IPARMS: • x, y (position) cells; resolution: 30 m DPARMS: • t (absolute time) sec; CSPICE ephemeris time; resolution: 1 sec/30 min

Table 6-1: LITA 2003 Planning Parameters

Feature	Description
Action Set	Mobile: • Drive Actions: one action for each of eight adjacent map cell neighbors Stationary: • Charge Actions: θ : solar optimal; Δt : 30 minutes • Hibernation Actions: low power; θ : solar optimal; Δt : 60 minutes
Constraint Set	Max. Slope, Daytime, Nighttime, Direct Sun Line-Of-Sight
Mission Specification Set	± 15 minute start time interval Start position, energy Single via point goals (no goal actions) Final goal energy
Planning Details	ISE Mode: BESTPCOST Objective function: composite path length and energy $h = nE_{max} + \sum_{i=1}^{n} \Delta e_i$ Better: if $t_1 < t_2$ Dominates: never Re-Planning: state update

6.5 Results 2003

Over April 17 through 20 and April 24 through 26, TEMPEST generated 27 plans and 83 re-plans. These experiments indicate qualitatively and quantitatively that TEMPEST planning sought longer than minimum length routes to avoid costly regions of the state space. Other analysis indicates that TEMPEST produced plans that were mildly unstable with respect to specific route, but exhibited clear arrival time stability. Most notably, TEMPEST enabled one traverse of over 1 kilometer, and many traverses of several hundred meters.

6.5.1 Path Length

TEMPEST's grid representation of position state interferes with the ability to produce shortest-distance paths, as described in Section 4.5. Recall that for a given ratio of Δx to Δy, the minimum increase in path length above the Euclidean distance is given by the representation factor f_R. Figure 6-4 examines the ratio of plan distance to Euclidean map distance for plans generated on April 20 through April 26 of the 2003 experiment to determine whether the grid representation was dominant in extending path length beyond the minimum. The horizontal axis spans the range of the absolute value of $\Delta x/\Delta y$, representing an East-West heading on the left, a Northeast-Southwest or Northwest-Southeast (diagonal) heading at the center, and a North-South heading on the right. The vertical axis spans the range of the plan distance D_{plan} divided by the Euclidean or map distance D_{map}. From Section 4.5, recall that this ratio is equal to the multiplication of the representation factor f_R and the avoidance factor f_A. The curve at the bottom of the plot shows the representation factor for the range of ratios of Δx and Δy. Therefore, the ratio of the point value to the curve value beneath it is the avoidance factor f_A .

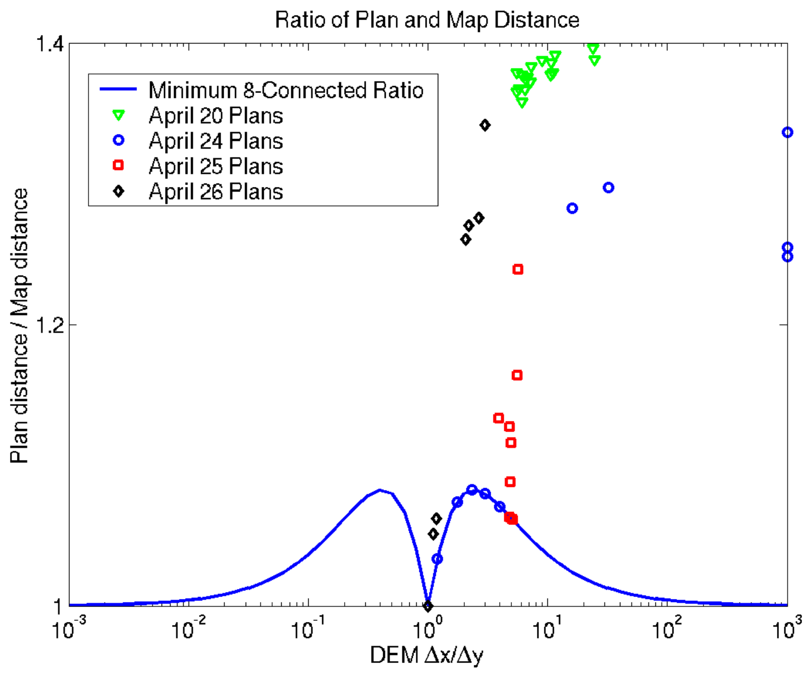
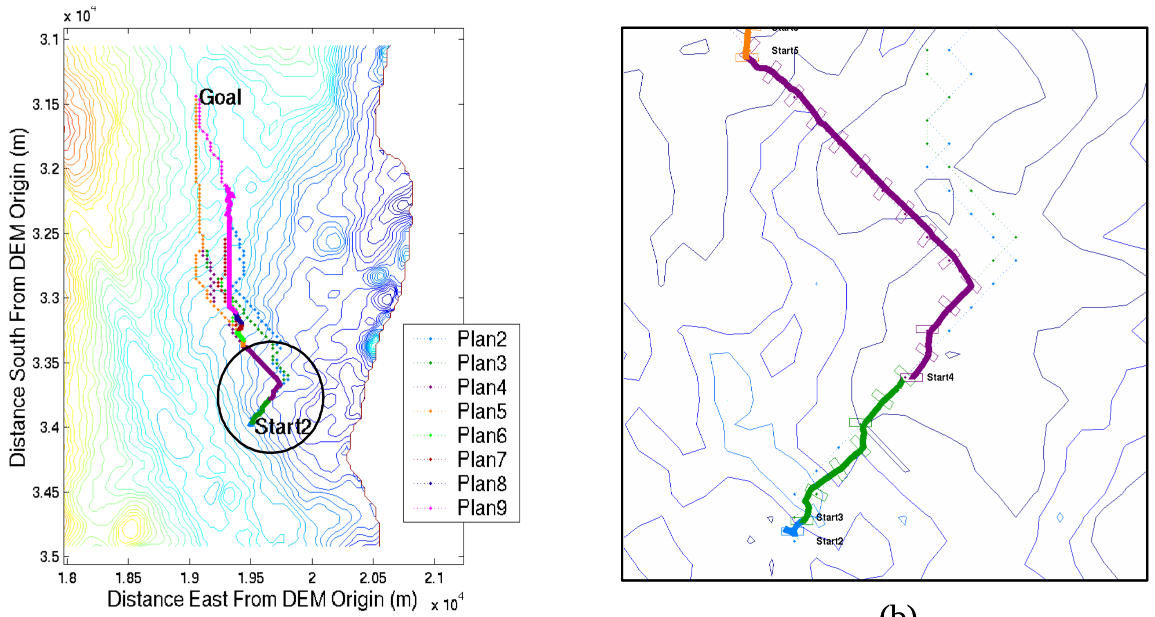


Figure 6-4: Avoidance Factor and Representation Factor for LITA 2003: The points suggest that avoidance was often dominant in determining path length.

Observe that all the plans fall barely to the right of center, confirming that routes traveled principally in a North-South direction, but with a slant from Northwest to Southeast. More interestingly, though several points fall very close to the minimum curve, most paths are much longer. This indicates a large avoidance factor and suggests that the eight-connected representation was *not* the principal contributor to path length extension for most of the plans. For the paths not near the minimum eight-connected curve, obstacle avoidance, energy cost minimization and constraint satisfaction contributed to path length extension, often significantly.

6.5.2 Large-Scale Terrain Avoidance

TEMPEST demonstrated large-scale hazard avoidance on several occasions. The planning for April 25 suggests subtlety. Figure 6-5a shows the sequence of plans and executed paths for the day. At first glance, the initial northeast heading taken by the plans is mysterious. Why did the planner force this detour rather than a more direct route to the goal? The answer appears to lie in slope avoidance. By plotting the same path over a contour map of the magnitude-of-gradient (slope) field (see Figure 6-6b), we observe that the path avoids steeper slopes to its left, and then turns toward the goal at a break in this higher slope region.



(a) (b)

Figure 6-5: Plan and re-plan routes from April 25 on an elevation contour map, and a close-up with contours of constant slope. The initial plans seem to have located a break in steeper slopes.

The most interesting, yet greatest failure in terrain avoidance occurred on April 18 (see Figure 6-6). In a plan to travel to the southern end of the area of operations, TEMPEST dictated a traverse near the rim of the large fault running in a primarily North-South direction to the West of base camp. According to observers near the robot on this day, waypoint goals dictated travel down precariously steep slopes on the West side of the fault ridge. This motion prompted the team to abort autonomous travel at this point. Errors in map registration with respect to the terrain were likely responsible for causing the problem.

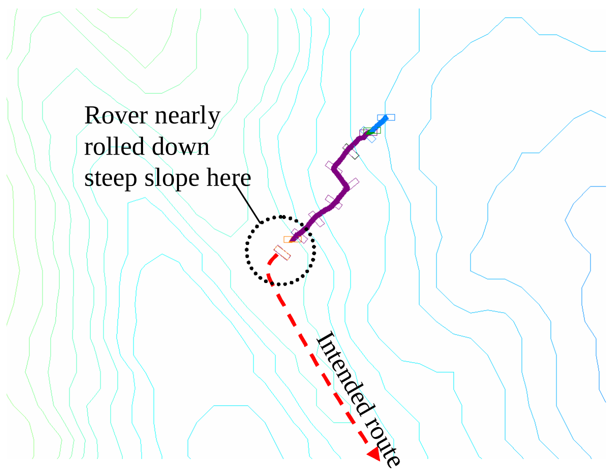


Figure 6-6: TEMPEST placed Hyperion very close to a hazardous slope on April 18. Map registration errors or position uncertainty may have been to

6.5.3 Energy Efficiency

In contrast to previous planning experiments in the Arctic [70], evaluating the plans from the Atacama field experiment proves to be difficult. For Atacama experiments, Hyperion's solar array was horizontal. This removed the strong coupling between the direction of travel and solar power in Hyperion's Arctic configuration, allowing the rover much more freedom of motion and schedule with little or no penalty. Furthermore, the solar flux in the Atacama was sufficiently high in April, during daylight, to sustain the highest-power operations indefinitely. Shadows only occurred very near sunset, so only intersected paths when operations were coming to a close. The planning models verify this - TEMPEST executed plans never included Charge or Hibernation actions. Finally, due to an undiscovered software bug, the telemetry logs did not log TEMPEST plan messages. Alternate records of plans, used to reconstruct plans from April 20 and later, did not include the battery energy variable of the plans. Unfortunately, this lack of data prevented determining when and where planning predicted energy-rich and energy-poor conditions.

6.5.4 Plan Monitoring and Re-Planning

A primary goal of the field experiment was to test re-planning in the context of rover operations and plan stability. As mentioned earlier, TEMPEST called upon state update re-planning. The Health Monitor provided simple plan execution monitoring, and was the sole trigger of re-planning.

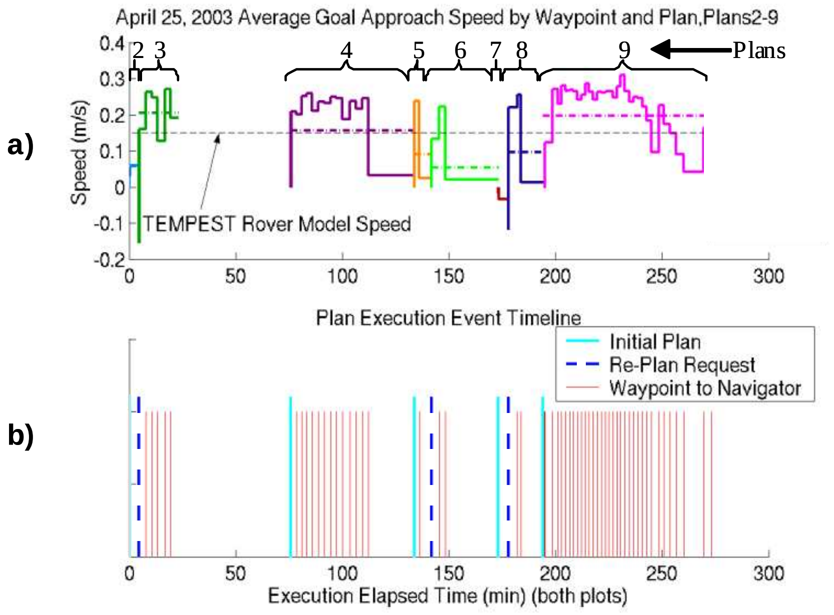


Figure 6-7: Rover Average Speed vs. Re-Plan Frequency. Operational delays often caused the HM to trigger re-planning. The solid lines in a) show average rover speed over a Drive action. The dashed lines are the average rover speed over the particular plan or re-plan execution. Speeds below the TEMPEST rover model speed (plans 2, 5, 7) caused re-plan events, shown in b). Blank regions in plot b) are human-designated suspensions of operation to enact manual fault recovery.

As position state estimation was quite accurate, the major cause of re-plan requests was deviation of average rover speed from the rover model, shown in Figure 6-7 for plans executed on April 25. The figure illustrates the connection between rover speed and HM re-planning requests[1]. Figure 6-7a plots speed as a function of time. The TEMPEST rover model speed is the constant, thin dashed line. The series of numbered brackets indicates the time spans for the execution of successive plans and re-plans. The solid traces for each plan show the average rover speed over the execution of each Drive action. Note that on several occasions, Hyperion stopped for long periods of time (end of plans 4, 5, 6, 8). These were not due to Charge actions, but reflect periods where the rover encountered irrecoverable faults and could not continue executing the plan. The dash-dot traces for each plan show the average rover speed over the entire plan or re-plan execution. Note that for plans 3 and 9, the average rover speed over a plan exceeds the speed assumed by the TEMPEST model, and in plans 2, 5, 6, 7 and 8, coinciding with faults that stopped the rover, the speed is much lower than predicted by the model.

Meanwhile, Figure 6-7b shows the timing of important plan execution events, denoted by vertical lines. The long solid lines correspond to TEMPEST initial planning runs, and the long dashed lines are re-plan events. The shorter, thinner lines correspond to when the Mission Executive sent waypoints to the Local Navigator. TEMPEST was manually terminated several times during the day after long operational delays (after plans 3, 4, 6 and 8). However, it is clear from plans 2, 5 and 7 that re-plans correlate well with periods of slow average driving speeds. The figure also underlines a logic error in the Health Monitor that overlooked faster-than-expected rover speed for re-planning. In no case does faster-than-predicted rover speed trigger a re-plan (see plans 3 and 9).

6.5.5 Plan Stability

Plan stability is determined by the degree to which plans vary in response to evolving initial rover state during a mission execution. Stable planning yields few changes in route or schedule with minor deviations from the current plan, and yields predictable changes for greater deviations. Unstable planning results in erratic behavior. A planner that exhibits stability enables mission operators to better predict the range of possible plan solutions without a exhaustive check. Whether planning is stable would also influence whether and how a planner might be integrated with other planners as a component within a greater autonomy software architecture. For example, if re-plans typically entail a total re-specification of the mission timeline, it might not be computationally practical to plan beyond the first action. Stable planning might permit a longer projection.

1. The gaps in data indicate time spans where autonomy was disabled by human operators to enact manual recovery actions from software or operational faults.

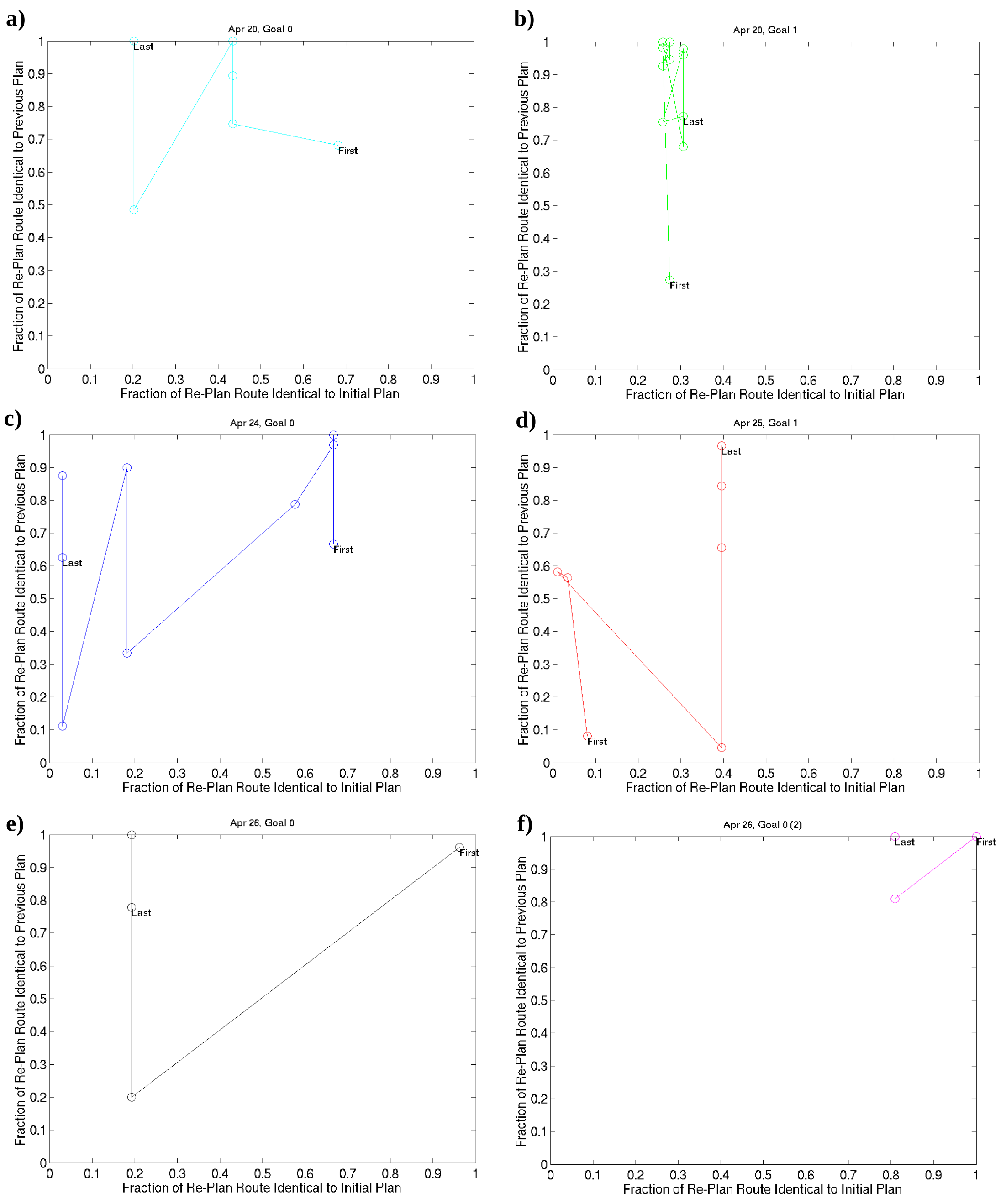


Figure 6-8: Route Stability: Most routes became more stable, as measured by progress to the upper left of the plot, with decreasing distance to the goal.

The degree to which *routes* are stable affects the degree to which the execution of the plan can be predicted. In a scenario where TEMPEST is used as an offboard planning tool, route stability would enable an engineering team to pre-validate TEMPEST plans without examining a wide range of contingency cases. Figure 6-8a) through f) illustrates plan route stability for the field experiment. Each frame depicts the route stability for a re-planning sequence in pursuit of a single goal, beginning with an initial plan and continuing with a number of re-plans. In each case, the horizontal axis shows the fraction of the re-plan waypoints that are identical to the *initial* plan (F_i). The vertical axis shows the fraction of the re-plan waypoints that are identical to those from the *previous* plan (F_p). The markers on the traces correspond to results from specific re-plans. The traces begin at the enlarged markers, the first re-plan, and proceed in chronological order. It follows that all traces begin on the line $F_i = F_p$, since for the first re-plan, the previous plan is also the initial plan.

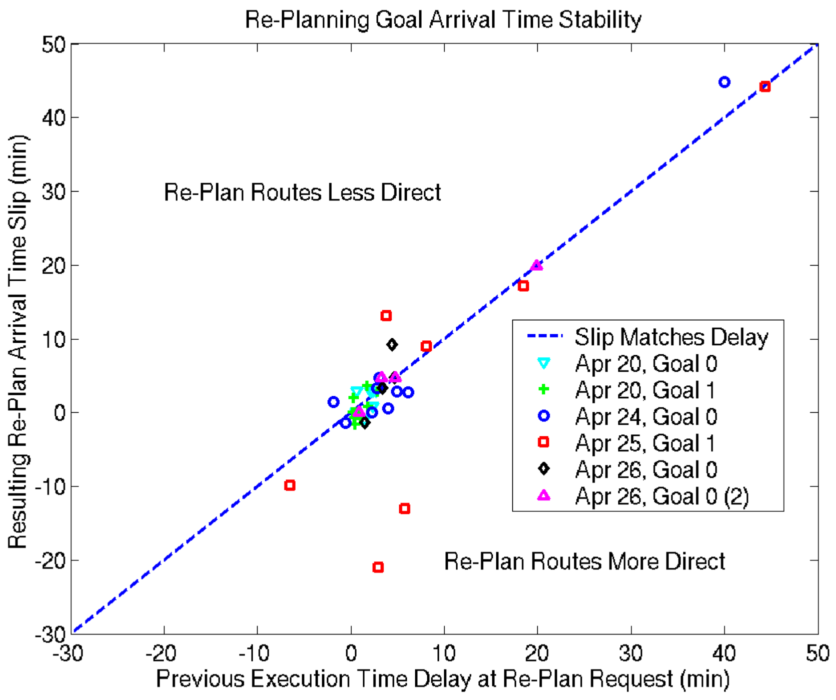


Figure 6-9: Arrival Time Stability: Changes in arrival time in re-plans correlate well with deviations from the previous plan during execution.

Observe that for all but one trace, the endpoint falls generally left and above the starting point. One can infer that for these cases, re-plans are initially unstable but grow gradually more stable as plan execution progresses. This seems to make intuitive sense. With the greater freedom that comes with a large distance between start and goal, ISE finds a number of plans of similar cost but with differing routes. Subtle changes in initial conditions may cause substantial route variations. However, as the distance to the goal shrinks, the freedom is reduced, leading to greater stability.

The exception is the plan sequence from April 25 (Figure 6-8d), whose first re-plan shares fewer than 10% of the initial route's waypoints. Successive re-plans deviate even more from the initial plan at first, but then return to match about 40% of the remaining plan. Figure 6-3a may help clarify what is happening in this case. The plans seem to alternate between two general routes over the last 1/2 of the traverse. Plan 2 (the initial plan shown) takes the right fork, Plan 3 (the first re-plan) the left. In the first half of the route, Plan 2 and Plan 3 are almost entirely distinct, but run very close to each other. In later planning instances, the plans settle on a variation of the right fork, increasing the fraction of the plan that is identical to Plan 2.

A planner exhibits arrival time stability when re-plans, in response to time deviations from an original plan, result in similar deviations in goal arrival time. Experiments indicate that TEMPEST planning are stable with respect to time. Figure 6-9 plots arrival time slip (vertical axis) against plan schedule slip (horizontal axis) for re-plans generated on April 20 through April 26. Each marker corresponds to a different re-plan instance. The dashed line falls where schedule slip exactly matches goal arrival delays. Re-plans falling above the dashed line are less direct then their predecessors, while re-plans below the line are more direct. Aside from a few outliers, the data seems to suggest a strong correlation between operational delays and schedule slips.

6.6 Field Experiment 2004

6.6.1 Objectives

In anticipation of full science operations for the 2005 field experiment, the principal autonomy goal for 2004 was to integrate science activities into operations. For TEMPEST this meant representing human-designated science goal actions within *Mission Specifications*, reasoning about the time and resource consumption of these activities in the scope of the global traverse plan, and enforcing temporal and energy constraints imposed on the completion of goals.

6.6.2 Zoe Rover

A new rover, Zoe, was developed to better integrate science instruments and to incorporate the lessons learned with Hyperion (see Figure 6-10). Its mass was 180 kg and its dimensions were 2.7 m long by 1.7 m wide, on par with the mass and size of Hyperion. It was far more capable of ascending steep slopes and crossing over rough terrain, and is designed mechanically to drive at a higher average speed. Enhanced computing onboard Hyperion enabled the Local Navigator to reliably avoid obstacles up to 1.0 m/s (3.6 km/hr), though system-level tests documented in this thesis were run at 0.5 m/s (1.8 km/hr).

Zoe's power configuration was more capable of collecting and storing solar energy than Hyperion. Zoe utilized a smaller solar array than Hyperion's (2.4 m^2) for supplying current to loads on the system and to charge batteries. The array's solar cells were triple junction cells, which provided a nominal efficiency of 24%, a substantial improvement over Hyperion's. The net effect between the size reduction and the efficiency enhancement was anticipated to be a 64% increase in solar power. Zoe's principal batteries for system-level tests were lithium-ion cells, designed for a maximum capacity of roughly 1340 W-hr of charge. The impact of these power upgrades was that in driving, Zoe was even more power-rich than Hyperion. However, with the addition of power-hungry science instruments enabled during science activities, it was not obvious whether the previous system would have been sufficient to sustain the rover in daily operations.

Increased navigational autonomy was a secondary goal of LITA 2004. To achieve a greater level of space relevance, rover state estimation no longer relied on GPS, but instead on a combination of wheel odometry, rate gyros, a sun tracker to enable absolute measurements of vehicle attitude, and a novel non-linear smoothing algorithm to update the position estimate history as new sun measurements are taken. This approach holds promise for the future, but was not functioning nominally during 2004 system-level tests. The impact to TEMPEST planning was that position state errors could no longer be maintained to below the resolution of the position state representation.

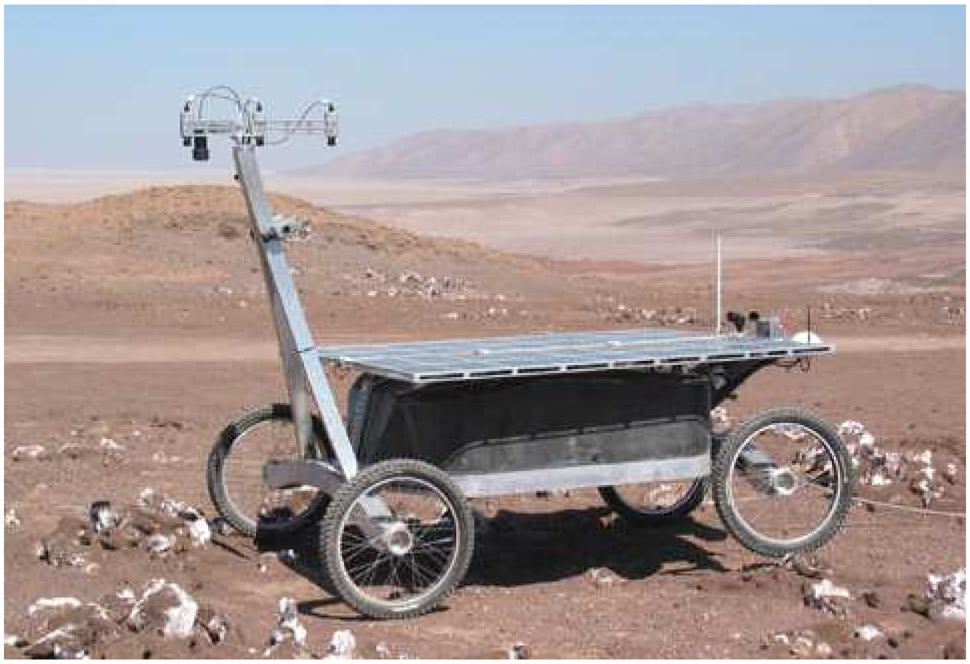


Figure 6-10: Zoe Rover

In contrast to Hyperion, Zoe had a substantial suite of science instruments, and could collect several types of measurements autonomously by the end of the 2004 field experiment. A stereo panoramic imager (SPI) comprised three cameras mounted on a pan/tilt head on the Zoe mast, enabling mono- and stereo panoramic image data sets. A fore-optic for Zoe's visual/near-infrared spectrometer (VNIR) was also mounted on the pan/tilt head, providing spectral data for regions of the panorama. Beneath the rover, a Fluorescence Imager (FI) could automatically deploy to activate and image the fluorescence of geologic or biotic materials. Optical cameras whose field of views covered the workspace of the FI provided context for fluorescence measurements.

6.6.3 Software Architecture

Hyperion's autonomy architecture was re-designed to better enable integrated science and navigation planning and execution and fault recovery. Zoe incorporated several new modules - a Rover Executive (RE), a replacement for Hyperion's Mission Executive, to coordinate the planning and execution of mission plans; a Goal Manager (GM) for pre-processing of goal specifications sent to TEMPEST and goal elaboration following TEMPEST planning; and an Instrument Manager (IM), an executive process to coordinate the execution of measurement activities.

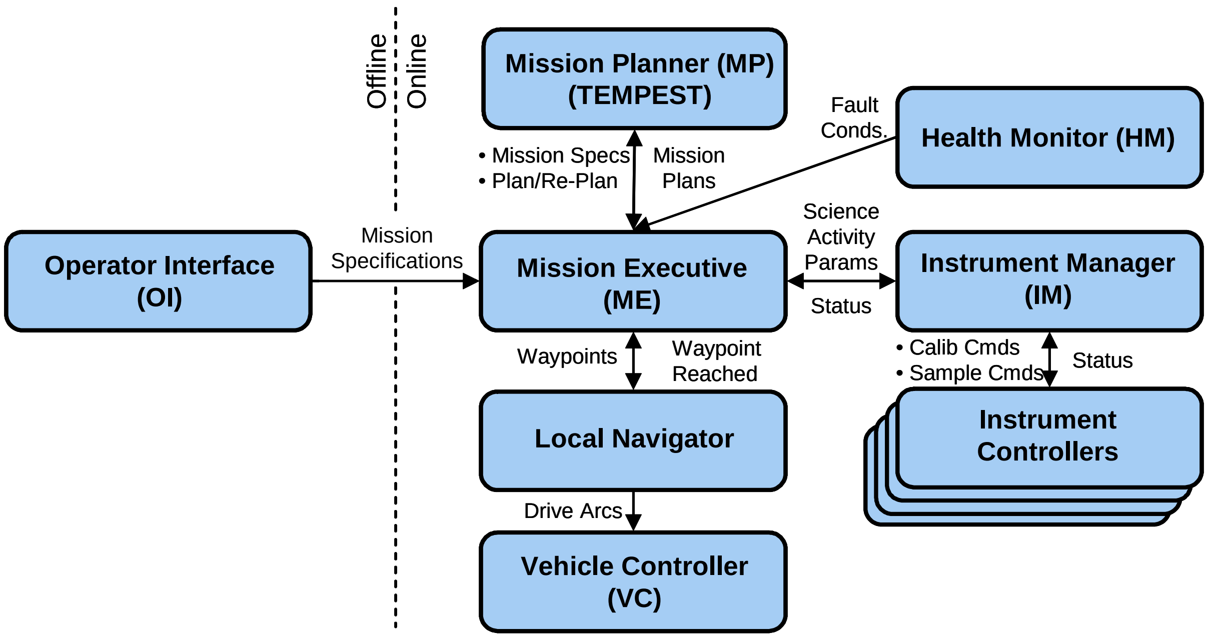


Figure 6-11: LITA 2004 Autonomy Software Architecture

The Rover Executive (RE) was developed under the IDEA architecture [45][15], and coordinated mission planning and execution. Unlike the previous Mission Executive, the RE aggregated logical models of each anticipated event and possible state transitions. It called upon the EUROPA planner [28][29] to perform both deliberative and reactive-scale temporal constraint checking on plans generated by the GM and TEMPEST, maintained a set of timelines to enforce temporal constraints on all planned activities, and enacted simple plan execution monitoring and recovery actions for common vehicle faults.

The RE coordinated the execution of plans encoded on the timelines. Drive actions were passed on to the Local Navigator as 18 m wide by 6 m deep goal regions similar to previous field experiments. Charge actions were executed directly by the RE by waiting the required duration. Science activities were passed to the IM, which managed the deployment of instruments and the execution of measurement sequences. For all activities, the RE monitored the execution timing of activities relative to the plan timelines. If either the execution completed prior to the earliest allowable time, or did not complete by the latest allowable time, the RE requested a re-plan from the GM and TEMPEST. The RE was *not* given the flexibility to anticipate the failure of an action to take early recovery steps.

The Goal Manager (GM) was designed as a pre- and post-processor for TEMPEST planning. As in 2003, goals for Zoe were specified exclusively by human operators. Under sequential goal planning described in Section 4.3.4, TEMPEST expected a fully-ordered sequence of goals. First, the GM elaborated goal activities, based upon goal activity parameters (e.g. size of panorama, number of pan-tilt steps), to predict appropriate time and energy allocations, and to convert coarse representations into more detailed specifications needed for execution.

Furthermore, the science and engineering team could not be expected to predict how many of the goals in the *Mission Specification* could feasibly be achieved within daylight hours, or which goal to remove if achieving the entire set was infeasible. Given a goal sequence, rewards assigned to each goal, and the latest time by which the mission must be completed, the GM used TEMPEST domain models and approximate energy constraints to estimate and select the highest-reward subset of goals achievable within the allotted time. The resulting subsequence of goals was sent to TEMPEST for planning. The GM returned elaborated plans to the RE for execution. Originally, the GM was intended to reduce the goal set in the event TEMPEST could not find a feasible plan. Time constraints in the software development schedule prevented this feature from being incorporated into the system.

6.6.4 Planning Approach

Table 6-2 summarizes the planning parameters used in the LITA 2004 experiments. To accommodate science activities, TEMPEST reinstated sequential goal planning, but unlike in the Arctic or in LITA 2003, with goal actions. TEMPEST was never intended to solve the general planning and scheduling problem typically solved by classical AI approaches (see Section 1.5). Because TEMPEST's principal role is to solve for traverse plans that satisfy temporal and resource constraints, the critical parameters for a goal action are its position, duration and resource (energy) consumption. For every science activity requested in a *Mission Specification*, the GM provided TEMPEST with duration and energy consumption upper bounds. For each of these activities, TEMPEST created a new science action using a

Generic Science action template, and added it to the *Action Set*. Other details, for example the hardware units designated for the activity or warmup and calibration procedures, were irrelevant to TEMPEST and left out of plan requests.

In preparing for possible 24-hour autonomy experiments in 2005 and beyond, one objective was to enable TEMPEST planning that would guarantee sufficient battery charge at the end of each to survive at low power overnight. Without knowing the completion time of a day's mission, it is difficult to assign goal battery energies that would enable overnight survival. To accommodate this, TEMPEST was augmented to enable time-bounded sequential goal planning, as described in Section 4.3.6. All *Mission Specifications* included an additional goal (with a null action) whose position was co-located with the final *requested* goal - an "End of Day" goal. To this goal, human operators assigned a legal time bounds corresponding to sundown and the battery energy required for night survival from that time. The time-bounded goal planning mechanism restricted plans to terminate at the position, within the time bounds and at the energy for the End of Day goal. The completion times and battery energies for *requested* goals remained free. With no specialized delay action in the *Action Set*, Charge actions would allow TEMPEST to insert delays into plans. To enable longer delays without added penalty, the LITA 2004 *Action Set* included additional, longer Charge actions.

The End of Day planning strategy, as stated above, did not result in the desired behavior. Ideally, plans would time-efficiently achieve all goals through Drive and Generic Science actions, and then loiter at the final goal position using Charge actions until satisfying the End of Day time bounds and energy. However, because TEMPEST planned in backward-chaining order, Drive actions were favored over most of the search, and Charge actions were only included in optimal plans to meet the conditions of the start time interval of the *Mission Specification*. This resulted in plans that began with Charge actions to accomplish the delay, followed by fast-as-possible Drive and Generic Science action sequences to achieve all the goals. Operationally, this embodied a risky strategy that required flawless execution of the traverse and science activities to meet the overall objective.

To remedy this situation, the objective function used in LITA 2003 was augmented with a third term that imposed "reward pressure." Human operators selected a reward pressure constant, expressed in units of power per unit reward (Watt-hours/reward), to be used over all planning segments. At each step in the search, the additional reward pressure cost was defined as the pressure power multiplied by the duration of the action multiplied by the reward remaining in future goals. The effect was to lightly penalize adding loiter actions towards the end of plans (the beginning of the search), but heavily penalize adding loiter actions towards the beginning of plans.

Table 6-2: LITA 2004 Planning Parameters

Feature	Description
World Model	Terrain: elevation, slope Ephemeris: CSPICE Solar Flux: constant value during daylight
Rover Model	Locomotion: simple force model (friction, gravity) Power: Solar array, re-chargeable battery Navigation Cameras: Field-of-view for sun blinding constraint
State Space	IPARMS: • x, y (position) cells; resolution: 30 m DPARMS: • t (absolute time) sec; CSPICE ephemeris time; resolution: 1 sec/30 min • g (goal completed Boolean)
Action Set	Mobile: • Drive Actions: one action for each of eight adjacent map cell neighbors Stationary: • Charge Action: θ : unspecified; Δt : 30, 60, 120, 240, 300 minutes • Generic Science: θ : unspecified; Δt , Δe : assigned by GM
Constraint Set	Max. Slope, Daytime, Nighttime, Direct Sun Line-Of-Sight, Sun In Camera
Mission Specification Set	± 15 minute start time interval Start position, energy Sequential via point goals with goal actions Final goal energy and time bounds
Planning Details	ISE Mode: BESTPCOST Objective function: composite path length, energy and reward pressure $h = nE_{max} + \sum_{i=1}^{n} \Delta e_i + P_{rew}\Delta t_i \sum_{j=G_{next}}^{G_N} r_j$ Better: if $t_1 < t_2$ Dominates: never Re-Planning: state update

6.7 Results 2004

An experiment toward the end of the 2004 field season illustrated mission-directed path planning in support of integrated science measurement and navigation. The *Mission Specification* required Zoe to perform Panorama actions (collect a full panorama image sequence) and Workspace Acquire actions (collect workspace camera images) at each of four goal locations, entailing a traverse of 2.74 km in map distance that terminated in a narrow valley. Figure 6-12 shows the terrain map and the initially-planned route for the traverse, which starts at the right of the map and progresses uphill into a canyon between two peaks. Table 6-3 summarizes the planning results.

6.7.1 Planning

In examining the distance and time factors in Table 6-3, one observes that increases in plan distance above the minimum were isolated to the eight-connected grid representation. The route diagram in Figure 6-12 confirms this graphically - the route follows horizontal, diagonals and vertical moves on the map grid rather than taking a direct path between goal positions. However, this is not to say that representation was at fault for increasing the distance. It seems apparent that TEMPEST used the degrees of freedom in the eight-connected path to avoid hazardous terrain (as illustrated in Figure 4-11). The final segment follows a path that avoids high slope areas by strategically alternating between diagonal and horizontal Drive actions. In many cases, the straight-line path between goals would have been inappropriate.

Table 6-3: Summary of October 18 Plan and Execution

Quantity	Planned
# of Goals	8 (4 positions)
Goal Spacing mean/min/max (m)	284 / 0 / 942
# of Actions	84 (76 Drive/0 Charge/4 Panorama/4 Workspace Acquire)
Representation Factor (f_R)	1.0728
Avoidance Factor (f_A)	1.0000
Loiter Factor (f_L)	1.0000
Distance (m)	2740
Duration (hh:mm)	01:08 (initial plan)

Figure 6-13 depicts the plan progress distance and minimum energy profile for the initial plan. The stair-stepping behavior of time-bounded sequential goal planning is plain in the distance plot. The plan allocates time for each goal to accommodate the Panorama and Workspace Acquire actions. The left dashed line in the distance plot is again the line of fastest possible approach under a speed of 1 m/s. The right dashed line is the line of slowest allowable approach to the goal, which for these runs was a factor 20 slower than the maximum speed. The intention is to provide ample time to allow for selecting circuitous routes or to insert Solar Charge actions. This scenario required neither, as reflected in the avoidance factor and loiter factor. The distance plot shows steady progress between each goal at a slightly lower slope than the line of fastest approach, reflecting the extra distance due to eight-connected travel. Unfortunately, this plan does not demonstrate the End Of Day goal mechanism - the End Of Day time bounds were set to open at the time of the start time of the plan, and close just before sundown.

Looking at the energy plot, the plan satisfies the initial condition of 200 W-hr and reaches the final goal energy requirement of 50 W-hr. TEMPEST models predicted no trouble in achieving the plan from a power perspective. The plan predicts that the rover could start with an empty battery and could remain fully discharged along the first two segments and still reach the goal target energy. It does predict a small, non-zero requirement at the start of the fourth segment. The plan terminates by rising to the target final goal energy.

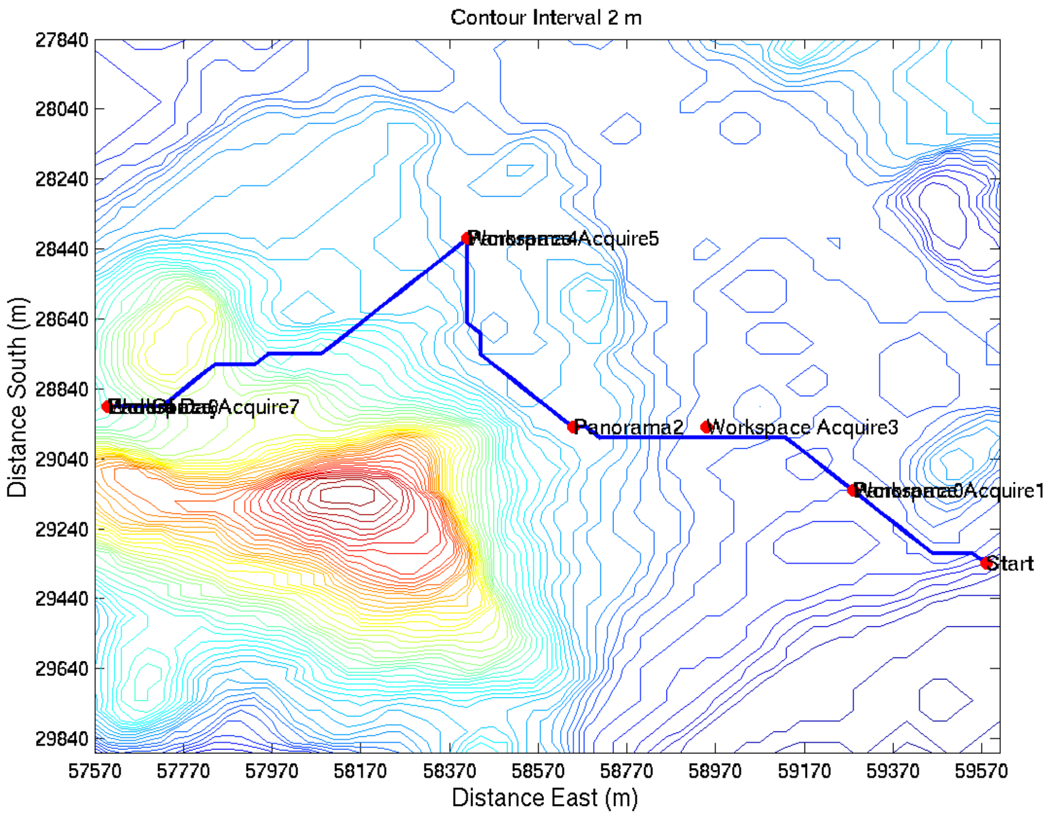


Figure 6-12: LITA October 18 2004 Route and Terrain Map

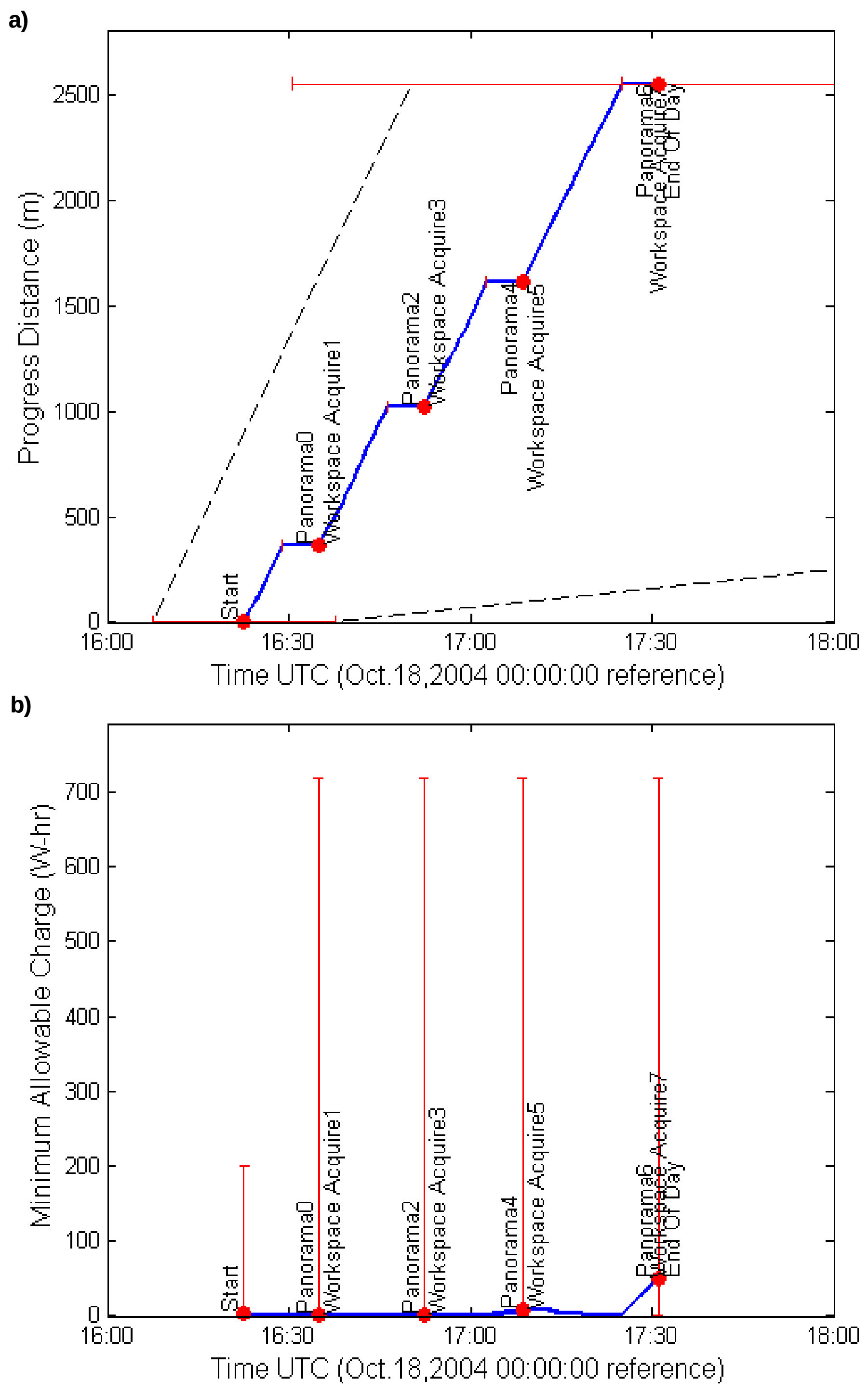


Figure 6-13: LITA October 18 2004: a) Progress Distance; b) Minimum Required Battery Energy

6.7.2 Execution

Global map registration was an unanticipated difficulty with the October 18 experiment. Prior to the experiment, the field team collected many GPS-derived ground control points that would allow cartographers at the US Geological Survey to associate absolutely-referenced positions with specific locations in unregistered elevation data collected from space[1]. Using those control points, the USGS provided the team with the map of the terrain that was intended to be referenced absolutely to Earth coordinates. A globally-referenced map would have allowed the team to initialize the rover state estimator with a correct map position using GPS. However, in initial tests, the team discovered that GPS measurements converted to map coordinates indicated substantial map registration errors (hundreds of meters in translation, and unknown errors in rotation). Matching GPS measurements of landmarks to salient elevation features in the map, the team attempted to better register the map to the Earth with translation. Closer examination of the translated map seems to indicate the attempted corrections were also in error.

Through the first three segments of the mission, Zoe exhibited reasonable navigational behavior. However, during the final segment, in attempting to enter the canyon area, Zoe attempted to follow a path that was farther south of the canyon opening than was suggested in the plan. Its course took it to the base of the large hill depicted in Figure 6-12, where it struggled to find traversable terrain on steep slopes through a network of water drainages. The preliminary judgment is that map registration errors prevented Zoe from entering the canyon at the correct point.Unfortunately, with mis-registered maps, the GPS "ground truth" from Zoe is of ambiguous value. It provides Zoe's absolute position on Earth, but does not yield Zoe's true path through the terrain model.

6.8 Discussion

These experiments highlight a number of important distinctions from experiments conducted in the Arctic. First, the mid-latitude environment presents a different set of challenges to mission-directed path planning. In polar summer, the sun never sets, but its low elevation angle does not favor a rover with a horizontal solar array. At mid-latitude, the sun rises and sets, but enables a rover to travel confidently during the day, under solar power, with a solar panel that is horizontally mounted and unarticulated. Daytime energy management in a polar environment demands a mechanical or navigational strategy, whereas daytime energy management at mid-latitude is not as much a significant challenge. From the perspective of demonstrating *persistent* operations on a planet, the two experiments were quite different. Arctic experiments demonstrated operations over 24 hours; Atacama experiments started well after dawn, occasionally extended until dusk, but never continued at night. Consequently, the LITA experimental planning and execution results presented here do not present the energy management challenges that might have arisen in enabling 24-hour operations.

1. Elevation data derived from imagery from the ASTER instrument aboard the Terra spacecraft.

TEMPEST's performance in LITA experiments demonstrated several new strengths above what was shown in the Arctic:

Terrain Avoidance: TEMPEST planned paths that avoided high terrain slopes. In a planetary setting, with a priori data, a robot using TEMPEST could anticipate large-scale terrain from "over the horizon" and take measures to avoid slope hazards from a distance. Current path planners for planetary exploration cannot consider terrain beyond their sensor horizon.

Integrated Science, Energy Management and Navigation: In the LITA 2004 experiment, TEMPEST coordinated mission and navigation activities effectively. It integrated naturally with other software modules - the Goal Manager, and the Rover Executive - to create plans that achieved the navigation goals of the mission and accommodated the requirements of science activities.

Effective Online Re-Planning: TEMPEST re-planning enabled far greater navigational autonomy than is possible by planning once in advance. A rover that can adapt its plans to unanticipated changes will be able to continue operating effectively without human intervention. In response to requests from plan monitoring modules, TEMPEST re-planned in fractions of the time required for initial planning. Re-planning periods rarely caused the rovers to halt for more than a fraction of a second.

The LITA experiments also uncovered challenges for future mission-directed path planning research:

Mission Re-Scoping: TEMPEST could not alter the scope of a mission in response to evolving state and environmental conditions. If operational delays are too significant, TEMPEST may not be able to find a feasible plan that meets time constraints. Conversely, if operations go more quickly than anticipated, TEMPEST cannot add more goals to the mission plan to take advantage of the situation. Deviations from expected behavior in position state and energy state can cause similar problems.

No Planning for Uncertainty: The Atacama again stressed the need to consider uncertainty in planning. Experiments in 2003 suggest map registration was responsible for a near disaster with Hyperion (see Section 6.5.2). Map registration in 2004 caused Zoe to struggle with navigation into a canyon. Also, time uncertainty was a problem. TEMPEST models only consider the nominal rover behavior. TEMPEST cannot anticipate the effects of operational delays, as often happen in the course of experiments, and more importantly, in planetary operations.

7. Conclusion

This thesis concludes that mission-directed path planning achieves a significant, practical advance in planetary rover autonomy, and enables a new, challenging class of planetary surface rover missions.

The research is significant because it extends path planning beyond local obstacle avoidance to time, resources and mission objectives and constraints - issues recognized by the space mission planning community to be of critical importance. Judging from MER, future missions will also seek to investigate regionally distributed targets, and may baseline years of operation. The greater ambition for autonomous regional exploration will require a commensurate sophistication in navigation and activity planning. Mission-directed path planning could automate path selection for regional exploration. With less manual planning to be done, missions would require far fewer operations staff and be correspondingly far cheaper. Greater robot autonomy would also reduce the frequency of decisions that require human intervention, resulting in less wasted time and greater return for each operational day. This research supports the ambition for cheaper, more efficient surface exploration.

This research demonstrates a practical solution to mission-directed path planning. TEMPEST derives plans that exhibit sensible navigation behaviors under complex interactions between terrain, time, resources and constraints. The approach combines models of the world, rover, relevant actions and constraints imposed on them, and mission objectives. Incremental search enables efficient search for optimal paths over three or more dimensions, and under global constraints. It offers efficient re-planning mechanisms to repair plans in response to unexpected state excursions and measurements of the local environment. Operating on the Hyperion and Zoe robots, TEMPEST operated efficiently and effectively in conjunction with automated local navigators, science planners and executives. In planning traverses of several hundred meters several hours in duration, TEMPEST spent on the order of ten minutes. Frequent re-plans thereafter caused only minor, and often imperceptible delays in progress.

Mission-level path planning enables a new class of planetary surface missions. Experiments demonstrate its utility in a number of specific scenarios: missions with overnight hibernation contingencies; polar exploration under sun-synchronous navigation; and missions that conduct widely distributed sampling to characterize regional variations. More generally, this new approach best addresses missions that operate in highly-variable, complex lighting and power, and missions that regularly interleave focussed stationary activities and extensive traverses. Chapter 1 introduces two planetary mission scenarios for which mission-directed path planning would certainly be an enabling technology.

Time, resources and mission objectives must factor into route selection to support the global needs of the mission. Without reasoning about these factors, planners must make conservative assumptions about legal operating ranges to guarantee a vehicle's safety. Imposing broad limits on operations can severely restrict the productivity of a robot. Occasionally, the conservatism required to guarantee rover safety disallows all operations. Deeper reasoning, through mission-directed path planning, allows a rover to take advantage of time and resource opportunities if they exist, and enables a measured level of protection against hazardous conditions when they arise.

7.1 Contributions

This research develops the most comprehensive global planner for planetary rovers to date.

Prior planetary mission planning has sought to optimize paths to avoid obstacles over traverses on the order of 100 meters. This new work achieves multi-kilometer planning, temporal and resource planning and interleaving of traverse and mission activities.

This research creates the first planner to optimize path selection for a non-monotonic resource.

The planning developed here incorporates resource collection as well as consumption to solve problems that are infeasible without recharge or refueling. Other path planners rely on monotonic resource models that cannot represent recharging. Non-monotonic resources, like battery energy, fuel or onboard memory, are commonplace in space, military, transportation and many other applications.

The research solves path planning in spatial, temporal and resource bounds using a non-hierarchical, resolution-optimal method.

The non-hierarchical approach developed in this research enables coupling between the spatial, temporal and resource state variables and solves for globally-optimal plans. Other path planners are sub-optimal or incomplete because they commit to a spatial path in one operation and then select a velocity or power profile to avoid time-varying obstacles and meet other constraints.

The thesis extends planetary rover path planning into the realm of more general mission planning.

The plans generated in the approach developed here consider the time and energy expense of mission goals and obey mission constraints. As demonstrated in field experiments in this research, a mission-directed path planner can act as a simple mission planner for a robot. If acting in support of a general mission planner, the planning developed in this research enables much tighter coupling between activities and traverses in a plan. Planetary rover mission planning has historically been the purview of classical artificial intelligence planning and scheduling approaches. They typically insert traverse plans into the mission plan that derive from a path planner that is incapable of reasoning about mission objectives.

The research successfully demonstrates mission-directed planning for solar powered robots in three planetary-relevant field experiments.

TEMPEST planning guided and sustained the solar powered Hyperion rover on a 6 km, 24-hour long polar sun-synchronous traverse. TEMPEST planning and re-planning enabled Hyperion to achieve several long-distance autonomous traverses in the Atacama Desert, including one over 1 km. TEMPEST created and maintained plans that allowed the Zoe rover to interleave a traverse of several hundred meters with several targeted panoramic and underbelly camera image sequences.

7.2 Perspectives

This thesis illustrates the power and limitations of incremental search as applied to time and resource-oriented path planning. In the positive, incremental search approaches offer the advantage of enabling guarantees of completeness and optimality. Conventionally, incremental search algorithms have been used to plan paths in two-dimensional spaces. Through the mission-directed path planning problem, this work illustrates the utility of incremental search for problems of greater than two dimensions. In enabling efficient representation of additional dimensions, the distinction between independent and dependent variables is a profitable segmentation of the state space. In effect, it collapses the representation of a very large space into the dimensionality of the independent variables, until the search dictates that other dimensions are important. Further, dependent variables can be treated at two resolutions, enabling efficiency mechanisms like resolution-based state pruning and state dominance. The resource optimization problem also demonstrates how resources can be correctly tracked and constrained outside the state space, and even optimized without adding the significant burden of an extra search dimension. Specifically, if state transitions and transition costs can be assumed to be independent of the resource variable, then that resource dimension can be represented within a composite objective function or as an auxiliary variable, at the cost of removing the guarantee of completeness. Reduction of search dimensions is the fastest means of reducing search time and space complexity, and in practice, the reduced-dimensionality planner readily produces solutions. Under these efficiency mechanisms, incremental search enables efficient planning and re-planning for complex domains, and yields provably resolution-optimal solutions.

Efficient re-planning is a major advantage of the incremental search approach. Re-planning enables a vehicle to compensate for unanticipated excursions away from an initial plan trajectory, and allows it to repair plans in light of new information. In most real-world mobile robot applications, particularly for planetary surface exploration, models of the world and the robot will be incomplete. A planning method that rapidly adjusts to on-the-fly measurements is essential for efficient robot operation.

This thesis also illustrates the limitations of an incremental search approach. Despite the array of available efficiency mechanisms, time and space complexity grows exponentially as dimensions are added to the state space. Specific to the mission-directed planning problem, time and energy are two of many interesting and important dimensions to the problem. One could easily envision problems where vehicle heading, multiple resources, and belief of state are equally important. In view of these larger, more general problems, it is not clear that incremental search is the correct approach.

Hierarchical navigation was proven highly effective in all three field experiment involving TEMPEST. The combination of a local navigator that senses the immediate environment and steers clear of rover-scale hazards with a mission-level path planner that reasons about the large-scale, time and resources is natural and powerful.

On the negative side, despite the array of available efficiency mechanisms available to incremental search, time and space complexity grow exponentially as dimensions are added to the state space. Specific to the mission-directed planning problem, time and energy are only two of many interesting and important dimensions that might be considered. One could envision problems where vehicle heading, multiple resources, and belief state are as important as time and energy. In view of so many additional variables, it is clear that incremental search cannot adequately address such problems.

TEMPEST proved vulnerable to a number of sources of uncertainty during the field experiments - principally time cost uncertainty, as demonstrated in the Arctic, and position state uncertainty as shown in both expeditions to the Atacama.

7.3 Future Work

Future work might fully characterize the benefit of mission-directed path planning, in comparison to standard spatial path planners, under varying terrain, lighting conditions and rover power configurations.

Mission simulations, controlled by a mission-directed planner and parameterized on terrain, lighting and rover energetics, could yield planning performance metrics in mission-relevant terms, for example time efficiency, energy efficiency, likelihood of success. Determining trends with respect to parameters would aid in establishing planning utility bounds for future missions.

Future work might investigate the use of rapid re-planning as a means of evaluating contingency branches in a meta-planning mode.

Fast re-planning could be leveraged to plan for hypothetical situations as easily as for actual ones. Research should evaluate the planning benefit in re-planning for hypothetical contingencies, and characterize the performance of re-planning under model updates not necessarily in the immediate vicinity of the rover.

Future work might characterize randomized and anytime algorithms that sacrifice optimality but enable efficient search over higher dimensional spaces, and evaluate them in the context of mission-directed path planning.

Research might determine whether randomized approaches can be designed to reliably generate reasonable, safe and mission effective solutions under probabilistic completeness and without the guarantee of optimality. Anytime algorithms could provide sub-optimal path solutions for high-dimensional spaces with known bounds on cost with respect to optimal. Adapting these search algorithms to mission-directed path planning might enable efficient planning over many additional variables, for example onboard memory, thermal state, or uncertainty parameters.

Research might develop an integrated approach for multi-scale navigation.

Unifying navigation at all scales might employ a common planning algorithm and encode a continuum of increasing representational granularity and decreasing re-planning frequency with distance from the robot. Unified navigation would extend the use of robot sensors to identify and characterize mid- and large-scale terrain features, foster a consideration of varying geometry, time, and resources at the local scale, and greatly streamline future rover software architectures.

Future work might enable efficient approaches to planning under uncertainty, with a consideration of risk sensitivity, in a mission-directed context.

Research must identify the sources of uncertainty most apt to disable mission-directed path planning, and evaluate current and develop new approaches to planning and sensing to diminish their effects. Furthermore, assessing plans in terms of risk, for instance by the variance in reward or cost, would enable a vehicle to select plans based on the evolving risk tolerance of the mission. Statistical methods are promising - they provide a natural, rigorous means of integrating sensing, planning and control, and have been successfully employed in an increasingly wide range of domains. Addressing uncertainty would enable more reliable planning for long-distance traverses that are particularly subject to errors in control, state and model accuracy.

References

[1] C. H. Acton Jr.,"Ancillary Data Services of NASA's Navigation and Ancillary Information Facility", *Planetary and Space Science*, 44 (1):65-70, 1996.

[2] M. Ai-Chang, B. Kanefsky, P. Maldague, P. Morris, K. Rajan, J. Yglesias, "MAPGEN: Mixed Initiative Planning and Scheduling for the Mars 03 MER Mission," *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics & Automation in Space (iSAIRAS-03)*, Japan, 2003.

[3] D. Bernard, G. Dorais, E. Gamble, B. Kanefsky, J. Kurien, G. Man, W. Millar, N. Muscettola, P. Nayak, K. Rajan, N. Rouquette, B. Smith, W. Taylor, Y. Tung, "Spacecraft Autonomy Flight Experience: The DS1 Remote Agent Experiment," *Proceedings of the AIAA Conference 1999*, Albuquerque, NM.

[4] D. Bernard, G. Dorais, C. Fry, E. Gamble, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. Nayak, B. Pell, K. Rajan, N. Rouquette, B. Smith, B. Williams, "Design of the Remote Agent Experiment for Spacecraft Autonomy," *Proceedings of the 1998 IEEE Aerospace Conference*, Snowmass, CO, 1998.

[5] J. Bobrow, S. Dubowsky, J. Gibson, "Time-Optimal Control of Robotic Manipulators Along Specified Paths," *International Journal of Robotics Research, Vol. 4, No. 3*, Fall 1985.

[6] J. Bresina, R. Washington, "Expected Utility Distributions for Flexible, Contingent Execution," *Proceedings of the American Association of Artificial Intelligence (AAAI-2000) Workshop: Representation Issues for Real-World Planning Systems*, Austin, TX, 2000.

[7] L. Bugayevskiy, J. Snyder, Map Projections: A Reference Manual, Taylor and Francis Group, 1995.

[8] R. Chatila, "Path Planning and Environment Learning in a Mobile Robot System," *Proceedings of the European Conference on Artificial Intelligence*, Orsay, France, 1982.

[9] S. Chien, R. Knight, A. Stechert, R. Sherwood, G. Rabideau, "Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling for Autonomous Spacecraft," *Workshop on Scheduling and Planning meets Real-time Monitoring in a Dynamic and Uncertain World, (IJCAI-99)*, Stockholm, Sweden, August 1999.

[10] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran, "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," *SpaceOps 2000*, Toulouse, France, June 2000.

[11] J. Cutts, S. Hayati, D. Rapp, C. Chu, J. Parrish, D. Lavery, R. DePaula, "The Mars Technology Program," *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics & Automation in Space (i-SAIRAS 2001)*, St-Hubert, Quebec, Canada, June, 2001.

[12] T. Estlin, F. Fisher, D. Gaines, C. Chouinard, S. Schaffer, I. Nesnas, "Continuous Planning and Execution for and Autonomous Rover," *Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space*, Houston, TX, October 2002.

[13] W. C. Feldman, S. Maurice, D.J. Lawrence, R.C. Little, S.L. Lawson, O. Gasnault, R.C. Weins, B.L. Barraclough, R.C. Elphic, T.H. Prettyman, J.T. Steinberg, A.B. Binder, "Evidence for Water Ice Near the Lunar Poles," *abstract of the Lunar and Planetary Science Conference XXXII*, Houston, TX, 2001.

[14] D. Ferguson, A. Stentz, "Planning with Imperfect Information," *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS 2004)*, Sendai, Japan, October 2004.

[15] A. Finzi, F. Ingrand, N. Muscettola, "Model-based Executive Control through Reactive Planning for Autonomous Rovers", *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS 2004)*, Sendai, Japan, October 2004.

[16] P. Fiorini, Z. Shiller, "Motion Planning in Dynamic Environments using Velocity Obstacles," *International Journal of Robotics Research, Vol. 17, No. 7*, July 1998.

[17] T. Fraichard, "Dynamic Trajectory Planning with Dynamic Constraints: a 'State-Time Space' Approach," *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 93)*, Yokohama, Japan, July 1993.

[18] R. Gaskell, J. Collier, L. Husman, R. Chen, "Synthetic Environments for Simulated Missions," *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, March 2001.

[19] S. Goldberg, M. Maimone, L. Matthies, "Stereo Vision and Rover Navigation Software for Planetary Exploration," *Proceedings of the 2002 IEEE Aerospace Conference*, Big Sky, MT, March 2002.

[20] M. Golombeck, D. Rapp, "Size-Frequency Distributions of Rocks on Mars and Earth Analog Sites: Implications for Future Landed Missions," *Journal of Geophysical Research, Volume 102, No. E2*, February 25, 1997.

[21] J.-P. Gonzalez, A. Stentz, “Planning with Uncertainty in Position: An Optimal Planner,” Carnegie Mellon University Robotics Institute Tech Report CMU-RI-TR-04-63, November 2004.

[22] A. Hait, T. Simeon, “Motion Planning on Rough Terrain for an Articulated Vehicle in Presence of Uncertainties,” *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 96)*, Osaka, Japan, November 1996.

[23] G. H. Heiken, D. T. Vaniman, B. M. French, eds., Lunar Sourcebook: A User's Guide to the Moon, Cambridge University Press, 1991.

[24] A. Howard, H. Seraji, E. Tunstel, “A Rule-Based Fuzzy Traversability Index for Mobile Robot Navigation,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2001)*, Seoul, Korea, 2001.

[25] K. Iagnemma, H. Shibly, A. Rzepniewski, S. Dubowsky, “Planning and Control Algorithms for Enhanced Rough-Terrain Rover Mobility,” *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2001),* St-Hubert, Quebec, Canada, June 2001.

[26] Jet Propulsion Laboratory website, “Update: Spirit and Opportunity”, http://marsrovers.jpl.nasa.gov/mission/status.html.

[27] Jet Propulsion Laboratory website, “Planetary Data System Standards Reference,” http://pds.jpl.nasa.gov/documents/sr/index.html.

[28] A. Jonsson, J. Frank, “A Framework for Dynamic Constraint Reasoning Using Procedural Constraints,” *Proceedings of the European Artificial Intelligence Conference*, 2000.

[29] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, B. Smith, “Planning in Interplanetary Space: Theory and Practice,” *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, Breckenridge, CO, April 2000.

[30] L. Kavraki, P. Svestka, J-C Latombe, M. Overmars, “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces,” *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, August 1996.

[31] A. Kelly, B. Nagy, “Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control,” *International Journal of Robotics Research, Vol. 22, No. 7,* July 2003.

[32] A. Kelly, O. Amidi, M. Happold, H. Herman, T. Pilarsky, P. Rander, A. Stentz, N. Vallidis, R. Warner, “Toward Reliable Autonomous Vehicles Operating in Challenging Environments,” *Proceedings of the International Symposium on Experimental Robotics (ISER ‘04),* Singapore, Malaysia, June 2004.

[33] O. Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,” *International Journal of Robotics Research, Vol 5. No. 1, p 60*, 1986.

[34] S. Koenig, M. Likhachev, “Improved Fast Replanning for Robot Navigation in Unknown Terrain,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002)*, Washington, D.C., May 2002.

[35] J. Kuffner Jr., S. LaValle, “RRT-Connect: An Efficient Approach to Single-Query Path Planning,” *Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA 00)*, San Francisco, CA, April 2000.

[36] S. LaValle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” Technical Report TR 98-11, Computer Science Department, Iowa State University, October, 1998.

[37] S. LaValle, S. Hutchinson, “An Objective-Based Framework for Motion Planning Under Sensing and Control Uncertainties,” *The International Journal of Robotics Research, Vol. 17, No. 1,* January 1998.

[38] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.

[39] S. Laubach and J. Burdick, “RoverBug: An Autonomous Path-Planner for Planetary Microrovers,” *Sixth International Symposium on Experimental Robotics (ISER 99)*, Sydney, Australia, March 1999.

[40] B. Logan, N. Alechina, “A* with Bounded Costs,” *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98),* Madison, WI, July, 1998.

[41] R. Loui, “Optimal Paths in Graphs with Stochastic or Multidimensional Weights,” *Communications of the Association of Computing Machinery, Vol. 26, No. 9*, September 1983.

[42] A. Mason, J. Burdick, “Trajectory Planning Using Reachable State-Density Functions,” *Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA 02)*, Washington, D.C., May 2002.

[43] A. Mishkin, J. Morrison, T. Nguyen, H. Stone, B. Cooper, B. Wilcox, “Experiences with Operations and Autonomy of the Mars Pathfinder Microrover,” *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, 1998.

[44] B. Murray, M. Malin, R. Greeley, Earthlike Planets, W. H. Freeman and Company, Copyright 1981.

[45] N. Muscettola, G. Dorais, C. Fry, R. Levinson, C. Plaunt, “IDEA: Planning at the Core of Autonomous Reactive Agents,” *Proceedings of the American Association for Artificial Intelligence (AAAI-01),* 2001.

[46] N. Nilsson, “A Mobile Automaton: An Application of Artificial Intelligence Techniques,” *Proceedings of the 1st International Joint Conference on Artificial Intelligence (IJCAI-99),* Washington, D.C., 1969.

[47] I. Nourbakhsh, M. Genesereth, "Assumptive Planning and Execution: a Simple, Working Robot Architecture," *Autonomous Robots, Vol. 3, No. 1*, 1996.

[48] C. O'Dunlaing, M. Sharir, C. Yap, "Retraction: A New Approach to Motion Planning," *ACM Symposium on Theory of Computing,* 15:207-220, 1983.

[49] D. Pai, L. Reissell, "Multiresolution Rough Terrain Motion Planning," *IEEE Transactions on Robotics and Automation, Vol. 14, No. 1*, February 1998.

[50] L. Pedersen, D. Smith, M. Deans, R. Sargent, C. Kunz, D. Lees, S. Rajagopalan, "Mission Planning and Target Tracking for Autonomous Instrument Placement," Proceedings of the IEEE Aerospace Conference, Big Sky, MT, March 2005.

[51] T. L. Perez, M. Mason, R. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots," *The International Journal of Robotics Research, Vol. 3, No. 1,* 1984.

[52] R. Richbourg, N. Rowe, M. Zyda, R. McGhee, "Solving Global, Two-Dimensional Routing Problems Using Snell's Law and A* Search," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation (ICRA '87),* Raleigh, North Carolina, March 1987.

[53] N. Rowe, D. Lewis, "Vehicle Path Planning in Three Dimensions Using Optics Analogs for Optimizing Visibility and Energy Cost," *Proceedings of the NASA Conference on Space Telerobotics*, Pasadena, CA, January 1989.

[54] N. Rowe, R. Ross, "Optimal Grid-Free Path Planning Across Arbitrarily-Contoured Terrain with Anisotropic Friction and Gravity Effects," *IEEE Transactions on Robotics and Automation Vol. 6, No. 5*, October 1990.

[55] N. Roy, W. Burgard, D. Fox, S. Thrun, "Coastal Navigation - Mobile Robot Navigation with Uncertainty in Dynamic Environments," *Proceedings of the 1999 IEEE Conference on Robotics and Automation (ICRA '99),* Detroit, MI, May 1999.

[56] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice-Hall Publishers, Upper Saddle River, NJ, Copyright 1995.

[57] K. Shillcut, "Solar Based Navigation for Robotic Explorers," Ph.D. thesis, technical report CMU-RI-TR-00-25, October 2000.

[58] Z. Shiller, Y. Gwo, "Dynamic Motion Planning of Autonomous Vehicles," *IEEE Transactions on Robotics and Automation, Vol. 7, No. 2,* April 1991.

[59] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja, K. Schwehr, "Recent Progress in Local and Global Traversability for Planetary Rovers", *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA 99),* Detroit, MI, May 1999.

[60] M. Slade, B. Butler, D. Muhleman, "Mercury Radar Imaging: Evidence for Polar Ice," *Science, Vol. 258, pp. 635-640*, 1992.

[61] R. Simmons, E. Krotkov, L. Chrisman, F. Cozman, R. Goodwin, M. Hebert, L. Katragadda, S. Koenig, G. Krishnaswamy, Y. Shinoda, W. Whittaker, P. Klarer, "Experience with Rover Navigation for Lunar-Like Terrains," *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS '95)*, Pittsburgh, PA, August 1995.

[62] R. Simmons, D. Apfelbaum, "A Task Description Language for Robotic Control," *Proceedings of the IEEE Intelligent Robots and Systems Conference (IROS 98)*, Vancouver, Canada, October 1998.

[63] A. Stentz, "The Focussed D* Algorithm for Real-Time Replanning", *Proceedings of The 14th International Joint Conference on Artificial Intelligence (IJCAI-9*5), Montreal, Canada, August 1995.

[64] A. Stentz, M. Hebert, "A Complete Navigation System for Goal Acquisition in Unknown Environments", *Autonomous Robots, Vol. 2, No. 2*, August 1995.

[65] A. Stentz, "Optimal and Efficient Path Planning for Unknown and Dynamic Environments", *International Journal of Robotics and Automation*, Vol. 10, No. 3, 1995.

[66] A. Stentz, "CD*: A Real-time Resolution Optimal Re-Planner for Globally Constrained Problems," *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, Edmonton, AB, Canada, July 2002.

[67] Z. Sun, J. Reif, "On Energy-Minimizing Paths on Terrains for a Mobile Robot," *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA '03),* Taipei, Taiwan, September 2003.

[68] R. Szczerba, D. Chen, "Using Framed Subspaces to Solve the 2-D and 3-D Weighted Region Problem," Department of Computer Science and Engineering Technical Report CSE 96-18, Notre Dame University, 1996.

[69] H. Takeda, C. Facchinetti, J-C. Latombe, "Planning the Motions of a Mobile Robot in a Sensory Uncertainty Field," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 6, No. 10*, October 1994.

[70] P. Tompkins, A. Stentz, W. Whittaker, "Mission Planning for the Sun-Synchronous Navigation Field Experiment," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 02)*, Washington D.C., May 2002.

[71] C. Urmson, M. Dias, R. Simmons, "Stereo Vision Based Navigation for Sun-Synchronous Navigation," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02)*, Lausanne, Switzerland, October 2002.

[72] C. Urmson, "Locally Randomized Kinodynamic Motion Planning for Robots in Extreme Terrain," Ph.D. thesis proposal, The Robotics Institute, Carnegie Mellon University, 2002.

[73] C. Urmson, J. Anhalt, M. Clark, T. Galatali, J. Gonzalez, J. Gowdy, J. Gutierrez, S. Harbaugh, M. Johnson-Roberson, Y. Kato, P. Koon, K. Peterson, B. Smith, S. Spiker, E. Tryzelaar, W. Whittaker, “High Speed Navigation of Unrehearsed Terrain: Red Team Technology for Grand Challenge 2004,” Carnegie Mellon University Robotics Institute Technical Report CMU-RI-TR-04-37, June 2004.

[74] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das, “The CLARAty Architecture for Robotic Autonomy,” *Proceedings of the 2001 IEEE Aerospace Conference*, Big Sky, MT, March 2001.

[75] Wellington, A. Stentz, “Online Adaptive Rough-Terrain Navigation in Vegetation,” *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA ‘04)*, New Orleans, LA, April 2004.

[76] M. Wellman, M. Ford, K. Larson, “Path Planning Under Time-Dependent Uncertainty,” *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, Montreal, Canada, August 1995.

[77] D. Wettergreen, B. Shamah, P. Tompkins, R. Whittaker, “Robotic Planetary Exploration by Sun-Synchronous Navigation”, *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 01)*, Montreal, Canada, 2001.

[78] D. Wettergreen, B. Dias, B. Shamah, J. Teza, P. Tompkins, C. Urmson, M. Wagner, W. Whittaker, “Experiments in Sun-Synchronous Navigation,” *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 02)*, Washington D.C., May 2002.

[79] D. Wettergreen, N. Cabrol, M. Deans, D. Jonak, A. Luders, F. Shaw, T. Smith, J. Teza, P. Tompkins, C. Urmson, V. Verma, A. Waggoner, M. Wagner, “Life in the Atacama: Field Season 2003 Experiment Plans and Technical Results,” Carnegie Mellon University Robotics Institute Technical Report CMU-RI-TR-03-50, October 2003.

[80] W. Whittaker, G. Kantor, B. Shamah, D. Wettergreen, “Sun-Synchronous Planetary Exploration,” *AIAA Space 2000 Conference and Exposition*, 2000.

Appendix 1: ISE Algorithm

This appendix describes the Incremental Search Engine (ISE) algorithm in greater detail than presented in Chapter 3. ISE search centers around a priority queue called the OPEN list. The first section describes how the OPEN list operates. The following section presents the ISE algorithm at the highest level, in terms of the ISE search modes: BEST-PCOST and BESTDPARMS. Each of these modes calls upon state expansion, whose details appear in the third section. An important feature of ISE is its ability to enforce and manage the dominance of some states over others. The final section in the appendix describes the piece of the algorithm associated with that management task.

A1.1 OPEN List

As with D*, ISE maintains a data structure, called the OPEN list, containing states prioritized for expansion. The OPEN list computes the path cost from states to the goals, and propagates information about changes to arc costs incurred during plan execution. The OPEN list propagates information by repeatedly expanding the highest priority state on the list. When a state is expanded, it is removed from the OPEN list, and all its child states are added to the OPEN list. States that are modified through cost increases or decreases are also placed on the OPEN list. All states have an associated tag function $t(X)$ that defines their OPEN list status. Tags hold one of three values: $t(X) = NEW$ if the X has never been on the open list, $t(X) = OPEN$ if X is currently on the OPEN list, and $t(X) = CLOSED$ if X was removed from the OPEN list.

For each state on the OPEN list, a key function $k(X)$ is defined to be equal to the minimum of $h(X)$ before a cost modification and over all values $h(X)$ assumed after X was placed on the OPEN list. The key function classifies states on the OPEN list into two types: RAISE states ($k(X) < h(X)$)and LOWER states ($k(X) = h(X)$). RAISE states propagate information about path cost increases, and LOWER states propagate information about path cost reductions.

States are placed on the OPEN list using their key value, which for LOWER states is the cost $h(X)$, and for RAISE states is the cost $h(X)$ prior to the cost increase. States are expanded in order of increasing $f^*(X, R) = k(X) + g(X, R)$, the optimal estimated path cost prior to a possible cost increase (note that for LOWER states, $f^*(X, R) = f(X, R)$). The intuition is as follows. RAISE states are not optimal, since $f^*(X, R) < f(X, R)$. Expansion allows them to re-route the path through lower-cost states. The value $f^*(X, R)$ of a RAISE state is a lower bound on the cost of LOWER states it can re-route to (it cannot activate a state that is less costly than the current optimum). Therefore, if all LOWER states' costs $f(X, R)$ exceed the $f^*(X, R)$ value of the RAISE state, it is better to expand the RAISE state first to possibly find lower-cost LOWER states. Note that if two OPEN states share the same $f^*(X, R)$ value, they are prioritized by their key values. This avoids creating cycles in state backpointers.

A1.2 Definitions

This section presents algorithmic details of ISE. The following two sections informally define several functions used in the ISE algorithm description, presented in section A1.3.

Table 1-1 lists the ISE application-specific functions that must be defined by the ISE user.

Table A1-1: Application-Specific ISE Functions

$better(X, Y)$	Encodes preferences of some DPARMS values over others. Given two states X and Y with equal IPARMS and path cost, this function returns $TRUE$ if X is preferred over Y, $FALSE$ otherwise.
$bneighbor(X)$	Returns the backwards neighbors of a state, using the backwards arc transition function $\beta(X)$. Given a state X, the function returns a list of states from which X is reachable.
$dominates(X, Y)$	Defines the conditions on DPARMS under which one state dominates another, according to the definition in section 3.2.3. Given two states with equal IPARMS, the function returns $TRUE$ if X dominates Y, $FALSE$ otherwise.
$domstates(X)$	Returns a list of states that might either dominate or be dominated by X, according to the definition provided in section 3.2.3.
$feasible(X)$	Defines the conditions on IPARMS and DPARMS under which states are feasible plan start states. Given a state X, this function returns $TRUE$ if X is a feasible start state, $FALSE$ otherwise.
$fneighbor(X)$	Returns the forwards neighbors of a state, using the forwards arc transition function $\Phi(X)$. Given a state X, the function returns of list of states reachable from X.
$resequal(X, Y)$	Determines when two states share the same resolution equivalence class. Given two states X and Y with equal IPARMS, this function returns $TRUE$ if X and Y are resolution-equivalent, $FALSE$ otherwise.

Table 1-2 describes several ISE domain-independent functions.

Table A1-2: Application-Generic ISE Functions

$b(X)$	A search back-pointer. Given a state X, it returns the next state Y on the optimal path to one of the goals.
$del(X)$	Returns $TRUE$ if a state X is marked for deletion, and $FALSE$ otherwise.
DELETE_STATE(X)	Deletes a state and associated memory from ISE.
$dom(X)$	Returns $TRUE$ if a state X is marked as dominated, and $FALSE$ otherwise.
GET_FMIN()	Returns the vector f_{min}, k_{min}, the path cost and key value corresponding to the highest-priority state X from the OPEN list, such that $f_{min} = k_{min} + g(X, R)$ and $k_{min} = k(X)$.
GET_MIN_STATE()	Returns the highest-priority state from the OPEN list, where priority is given to the state X with the lowest value of $f(X, R)$. It removes the state from the OPEN list and sets $t(X) \leftarrow CLOSED$ and $proc(X) \leftarrow TRUE$.
GREATEREQ(f_1, k_1 , f_2, k_2)	Compares two vectors $v_1 = f_1, k_1$ and $v_2 = f_2, k_2$ and returns $TRUE$ if $f_1 > f_2$ or ($f_1 = f_2$ and $k_1 \geq k_2$), and $FALSE$ otherwise.
INSERT_STATE($X, h(X)$)	Inserts a state directly into the OPEN list, with a path cost of $h(X)$.
LESS(f_1, k_1 , f_2, k_2)	Compares two vectors $v_1 = f_1, k_1$ and $v_2 = f_2, k_2$, and returns $TRUE$ if $f_1 < f_2$ or ($f_1 = f_2$ and $k_1 < k_2$), and $FALSE$ otherwise.
$modified(X)$	Returns $TRUE$ if a state is downstream (with respect to the search) of a set whose APARMS have been modified, and $FALSE$ otherwise.
$proc(X)$	Returns $TRUE$ if a state has been processed off the OPEN list (via GET_MIN_STATE()) at least once, and $FALSE$ otherwise.
$set(X)$	Returns the state set containing X. Given a state X, it returns the set of states sharing the same IPARMS as those of X.
$t(X)$	Supplies the OPEN list tag for a state. Given a state X, it returns NEW for states that have never entered the OPEN list, $OPEN$ if a state is currently on the OPEN list, and $CLOSED$ if the state has been processed and is currently not in the OPEN list.

A1.3 BESTPCOST and BESTDPARMS Modes

At the highest level, ISE operates in either of two modes, BESTPCOST and BESTDPARMS. The BESTPCOST mode finds the minimum cost path. The BESTDPARMS mode finds the "best" path below a user-designated upper bound in path cost, where better paths are defined using the application-specific $better(X, Y)$ function in Table 1-1. Both modes share the underlying search algorithm, composed principally of two functions -

EXPAND_NEXT_STATE and INSERT_STATE_CHECK. The two modes and these functions are described in detail in the following paragraphs.

The BESTCOST mode finds the minimum cost path. The BESTCOST algorithm appears in Table 1-3. Within an upper bound on iterations, ISE continues to expand states (L4), checking at each expansion whether the start set containing the robot start state R has been affected (L5). In lines L6-L7, ISE scans the updated $set(R)$ for states that meet the termination criteria. A state must be feasible, must have been processed off the OPEN list (not awaiting cost updates), and its cost must be lower than the least cost estimate on the OPEN list (unless the OPEN list is empty) and than the current optimal solution. The optimal plan begins from the lowest cost state of states that first meet these criteria. Given ISE prioritizes lowest-cost states for expansion, the first feasible state to be expanded also tends to be the start state of the optimal solution. ISE returns the optimal state, its path cost and whether or not the maximum number of iterations was exceeded.

Table A1-3: BESTPCOST Mode Algorithm

```
L1   $R \leftarrow$ start state , $S_{optimal} \leftarrow NULL$ , $i \leftarrow 0$
L2   while $i < TIMEOUT$
L3       $h_{optimal} \leftarrow \infty$
L4       $f_{min}, k_{min}, updated \leftarrow$ EXPAND_NEXT_STATE()
L5       if $updated = TRUE$ or $i = 0$ then
L6           for each state $x \in set(R)$ do
L7               if $feasible(x)$ and $t(x) = CLOSED$ and $(h(x) \le f_{min}$ or $f_{min} = -1)$ and $h(x) < h_{optimal}$ then
L8                   $S_{optimal} \leftarrow x$, $h_{optimal} \leftarrow h(x)$
L9       if $f_{min} = -1$ or $(h_{optimal} < f_{min}$ and $feasible(S_{optimal}))$ then return $S_{optimal}, h_{optimal}, TRUE$
L10      $i \leftarrow i + 1$
L11  return $S_{optimal}, h_{optimal}, FALSE$
```

The BESTDPARMS mode finds the solution with the “best” start state DPARMS that falls below a maximum path cost. In this mode, listed in Table 1-4, the objective function serves only to measure path costs against the maximum. In lines L13-L16, ISE continues to expand states until the OPEN list is exhausted or the minimum path cost on the OPEN list exceeds the cost upper bound. This ensures that all possible solutions at or below the upper bound cost are processed. ISE evaluates all states deposited into $set(R)$ as a result of the search. As in the BESTPCOST mode, the best state must be a feasible start state, must be CLOSED, and must cost less than any state remaining on the OPEN list. In contrast to BESTPCOST, the state need not be the least costly, but must fall beneath the cost upper bound. The search may generate many states satisfying these criteria. The $better$ function prioritizes the qualifying states. The search ends when no states remain within $set(R)$ below the cost upper bound. The optimal start state is the "best" from among the qualifying states.

Table A1-4: BESTDPARMS Mode Algorithm

```
L12  $R \leftarrow$ start state , $i \leftarrow 0$
L13  while $i < TIMEOUT$
L14          $f_{min}, k_{min}$ , $updated \leftarrow$ EXPAND_NEXT_STATE()
L15      if $f_{min} = -1$ or $f_{min} > MAXCOST$ break
L16      $i \leftarrow i + 1$
L17  $h_{best} \leftarrow \infty$ , $checked \leftarrow FALSE$
L18  for each state $x \in set(R)$ :do
L19      if $feasible(x)$ and $t(x) = CLOSED$ and ( $h(x) \le f_{min}$ or $f_{min} = -1$ ) and $h(x) < MAXCOST$ and
                    ( $checked = FALSE$ or $better(x, S_{best})$ ) then
L20          $S_{best} \leftarrow x$ , $h_{best} \leftarrow h(x)$ , $checked \leftarrow TRUE$
L21  if $i < TIMEOUT$ then return $TRUE$
L22  return $FALSE$
```

A1.4 State Expansion

Both ISE modes depend fundamentally on the function EXPAND_NEXT_STATE, listed in detail in Table 1-5. This function processes states off the OPEN list and communicates their effects, in terms of cost updates and dominance, on neighboring states.

The function begins by removing the lowest cost state X from the OPEN list. It returns an invalid cost vector if the OPEN list is empty. If the state is downstream of a modification to APARMS, lines L26-L30 designate the backwards (downstream) neighbors of X, whose paths contain X, for deletion and recycles them onto the OPEN list for future processing. This allows ISE to delete the old graph structure, created under the old APARMS, in preparation for creating graph nodes under the modified arc costs.

If $h(X) > k(X)$, then X is a RAISE state, and may not be optimal. In lines L31-L34, ISE checks all the forward neighbors of X to determine whether they might provide an alternate, cheaper path to the goal, with correspondingly lower $h(X)$. If so, the state's deletion status is unmarked, the path is re-directed, the state cost is re-computed to be the cost to the goal through Y, and the state is marked as a reduced RAISE state (L34).

RAISE states do not necessarily hold the same dominance relationships. If the state is a RAISE state at L34, then L35 re-checks the state's dominance relationship in the OPEN list, without re-inserting it. Then, for states that remain RAISE states, ISE re-creates the forward neighbor states that were previously dominated, and re-inserts them directly to the OPEN list for re-processing (L36-L39). Finally, if a RAISE state itself becomes dominated, ISE marks it for later deletion on line L40.

Following the previous reduction operations, the state could be either a RAISE or LOWER state. If $h(X) = k(X)$, then X is a LOWER state and by definition optimal. Lines L42-L49 search amongst the backwards neighbors of LOWER state X and propagate cost changes to them. Three conditions prompt ISE to re-insert these states to the OPEN list (L43). First, if the neighbor is a NEW state, it must be supplied with an initial cost. Second, if a path is directed from the neighbor to the state X, but the neighbor's cost is mismatched, then its cost must be re-matched to reflect the optimal cost through X. Third, if the path from the neighbor does not go through X, and the neighbor's cost is higher than if it *were* directed through X, its cost must be decreased to reflect a re-direction through X. In all three cases, if X is marked for deletion, so are the neighbors (L45). If not, the neighbor backpointers are directed through X (L47). Furthermore, in all cases the neighbor is re-evaluated for dominance and placed onto the OPEN list with its new, optimal cost (L48).

Table A1-5: EXPAND_NEXT_STATE Algorithm

```
L23  $X \leftarrow$ GET_MIN_STATE() , $R \leftarrow$ start state , $updated \leftarrow FALSE$ , $reduced \leftarrow FALSE$
L24  if $X = NULL$ then return $-1, -1$
L25  if $X = R$ then $updated \leftarrow TRUE$
L26  if $modified(X) = TRUE$
L27      for each state $Y \in bneighbor(X)$ do
L28          if $b(Y) = X$ then
L29              $del(Y) \leftarrow TRUE$ , $h(Y) \leftarrow \infty$ , INSERT_STATE($Y, h(Y)$)
L30              if $set(Y) = set(R)$ then $updated \leftarrow TRUE$
L31  if $h(X) > k(X)$ then
L32      for each state $Y \in fneighbor(X)$ do
L33          if LESS( $f(Y, R), h(Y)$ , $k(X) + g(X, R), k(X)$ ) and $h(X) > h(Y) + c(Y, X)$ then
L34              $del(X) \leftarrow FALSE$ , $b(X) \leftarrow Y$ , $h(X) \leftarrow h(Y) + c(Y, X)$ , $reduced \leftarrow TRUE$
L35  if $reduced = TRUE$ then DOM_INSERT_STATE($X, h(X), FALSE$)
L36  if $h(X) > k(X)$ then
L37      for each state $Y \in domfneighbor(X)$ do
L38          if $t(Y) = CLOSED$ then INSERT_STATE($Y, h(Y)$)
```

Table A1-5: EXPAND_NEXT_STATE Algorithm

```
L39             if $set(Y) = set(R)$ then $updated \leftarrow TRUE$
L40     if $dom(X) = TRUE$ then $del(X) \leftarrow TRUE$, $h(X) \leftarrow \infty$
L41     if $h(X) = k(X)$ then
L42         for each state $Y \in bneighbor(X)$ do
L43             if $t(Y) = NEW$ or ($b(Y) = X$ and $h(Y) \neq h(X) + c(X, Y)$) or ($b(Y) \neq X$ and $h(Y) > h(X) + c(X, Y)$) then
L44                 if $del(X) = TRUE$ then
L45                     $del(Y) \leftarrow TRUE$, $b(Y) \leftarrow NULL$
L46                 else
L47                     $del(Y) \leftarrow FALSE$, $b(Y) \leftarrow X$
L48                 DOM_INSERT_STATE($Y$, $h(X) + c(X, Y)$, $TRUE$)
L49                 if $set(Y) = set(R)$ then $updated \leftarrow TRUE$
L50     else
L51         for each state $Y \in fneighbor(X)$ do
L52             if $b(Y) \neq X$ and $h(X) > h(Y) + c(Y, X)$ and $t(Y) = CLOSED$ and
                        GREATEREQ( $f(Y, R), h(Y)$ ,  $k(X) + h(X), k(X)$ ) then
L53                 INSERT_STATE($Y$, $h(Y)$)
L54                 if $set(Y) = set(R)$ then $updated \leftarrow TRUE$
L55         for each state $Y \in bneighbor(X)$ do
L56             if $t(Y) = NEW$ or ($b(Y) = X$ and $h(Y) \neq h(X) + c(X, Y)$) then
L57                 if $del(X) = TRUE$ then
L58                     $del(Y) \leftarrow TRUE$, $b(Y) \leftarrow NULL$
L59                 else
L60                     $del(Y) \leftarrow FALSE$, $b(Y) \leftarrow X$
L61                 DOM_INSERT_STATE($Y$, $h(X) + c(X, Y)$, $TRUE$)
L62                 if $set(Y) = set(R)$ then $updated \leftarrow TRUE$
L63             else if $b(Y) \neq X$ and $h(Y) > h(X) + c(X, Y)$ then
L64                 INSERT_STATE($X$, $h(X)$)
L65     return   GET_FMIN(), $updated$
```

If the state is not a LOWER state, then it remains a RAISE state that may affect both is forwards and backwards neighbors. ISE searches the forward neighbors first (L51). If a forward neighbor does not direct a path through X and the path from X through the neighbor reduces the cost from X, and the neighbor is not awaiting processing (L52), then ISE re-inserts the neighbor into the OPEN list with its old cost (L53). For backwards neighbors (L55), three conditions prompt changes (L56 and L63). As with the backwards neighbors of LOWER states, if the backwards neighbor of the RAISE state is NEW, or if it directs a path through X but its cost does not match, its cost is updated and it is placed on the OPEN list after having its dominance relationships re-evaluated. Finally, if the path from the neighbor does not pass through X, and the neighbor's cost can be lowered by going through X (L63), then the state X itself is re-entered onto the OPEN list, since X is not necessarily optimal. This action is required to avoid creating a closed loop in the backpointers.

Function EXPAND_NEXT_STATE returns the vector the estimated path cost key value of the highest-priority state on the OPEN list, as well as a Boolean value signifying whether or not a state contained in $set(R)$ was modified before being placed back on the OPEN list.

A1.5 State Dominance Management

The management of dominance and "better" relationships among states is a key component of the ISE algorithm. Dominated states cannot legally be a part of any ISE solution. Similarly, during a BESTDPARMS search, states given lower priority according to the $better(X, Y)$ function cannot be a part of a solution. The difficulty is that during re-planning, dominating and "better" states may become too costly to be optimal, in which case the states they dominated become candidates for re-consideration. In the original implementation of ISE, states dominated by other states were maintained in their IPARMS sets in anticipation of that case. In practice, this led to excessive memory consumption for storage, since a majority of dominated states remained dominated during re-planning. The solution was to implement a function that manages these relationships, and deletes the states that are dominated. Just as in initial search, states can be re-generated in the event that their dominators grow too expensive.

Function DOM_INSERT_STATE$(X, h(X), flag)$ inserts a state X into the OPEN list after checking and asserting state dominance relationships between the state and other states in the OPEN list. A state X is not put on the OPEN list if it is dominated by other existing states, and states dominated by X are deleted from the list. If $flag$ is set to $TRUE$, the state may be put on the OPEN list; if $flag$ is set to $FALSE$, its dominance relationships are checked, but it is not inserted into the OPEN list.

At the outset, the new state is considered not to be dominated (L66). The function searches over all old states that might either dominate or be dominated by X (L67). If the new state X is resolution equivalent to another state, then the function determines which state the lesser state and can be deleted from memory. Lines L69-L74 check whether state X is either cheaper than the old state, or if equivalent in cost, whether the new state's DPARMS values are preferred over the other's. If proven inferior, the old state is deleted outright if it has never been processed, and marked as dominated and placed back on to the OPEN list if not. If the old state is not proven inferior, then the state X itself is deleted or marked dominated and placed on the OPEN list in the same manner (L75-L81). If X is deleted the function returns.

If new state X is not resolution equivalent to the old state, the function checks to see whether a dominance relationship exists between the two. If X dominates over the old state, and the new state has the same or lesser cost, then the old state is deleted or marked dominated and placed on the OPEN list (L83-L89). If the old state dominates over the new state, then the new state is processed accordingly and the function returns (L90-L97). If the new state has not been deleted or dominated, and the insertion flag is set, then the new state is inserted into the OPEN list.

Table A1-6: DOM_INSERT_STATE Algorithm

```
L66  $dom(X) \leftarrow FALSE$ , $sflag \leftarrow FALSE$
L67  for each state $Y \in domstates(X)$ do
L68      if $resequal(X, Y)$ then
L69          if $h(X) < h(Y)$ or ( $h(X) = h(Y)$ and $better(X, Y)$ ) then
L70              if $proc(Y) = FALSE$ then
L71                  DELETE_STATE($Y$) , $sflag \leftarrow TRUE$
L72              else
L73                  $dom(Y) \leftarrow TRUE$
L74                  if $t(Y) = CLOSED$ then INSERT_STATE($Y, h(Y)$)
L75          else
L76              if $proc(X) = FALSE$ then
L77                  DELETE_STATE($X$)
L78              else
L79                  $dom(X) \leftarrow TRUE$
L80                  if $t(X) = CLOSED$ and $flag = TRUE$ then INSERT_STATE($X, h(X)$)
L81              return
L82      else
L83          if $dominates(X, Y)$ then
L84              if $h(X) \leq h(Y)$ then
L85                  if $proc(Y) = FALSE$ then
L86                      DELETE_STATE($Y$) , $sflag \leftarrow TRUE$
L87                  else
L88                      $dom(Y) \leftarrow TRUE$
L89                      if $t(Y) = CLOSED$ then INSERT_STATE($Y, h(Y)$)
L90          else if $dominates(Y, X)$ then
```

Table A1-6: DOM_INSERT_STATE Algorithm

```
L91          if $h(Y) \leq h(X)$ then
L92              if $proc(X) = FALSE$ then
L93                  DELETE_STATE($X$)
L94              else
L95                  $dom(X) \leftarrow TRUE$
L96                  if $t(X) = CLOSED$ and $flag = TRUE$ then INSERT_STATE($X, h(X)$)
L97              return
L98  if $flag = TRUE$ then INSERT_STATE($X, h(X)$)
```

Appendix 2: Progress Distance

In temporal path planning, it is difficult to express spatio-temporal relationships in two-dimensional plots. Plots of position ignore temporal aspects. Plots of the individual spatial dimensions against time can be confusing. *Progress distance* captures important aspects of the spatial dimensions in a single parameter, and plots of progress distance versus time are useful tools for analyzing spatio-temporal plans.

To define the term, assume a plan intersects a fully-ordered sequence Γ of N goals, that the Euclidean distance between the start position S and the first goal G_1 is given by $\rho(S, G_1)$, and the distance between two adjacent goals G_i and G_{i+1} is given by $\rho(G_i, G_{i+1})$. The total minimum plan distance is then given by:

$$R = \rho(S, G_1) + \sum_{i=1}^{N-1} \rho(G_i, G_{i+1}) \qquad \text{A2-1}$$

During the course of execution, goals may be achieved by a robot and removed from the list of remaining goals. If the new start state is X and the first remaining unachieved goal is given by $next(\Gamma)$, then the minimum remaining distance is given by:

$$D_{rem} = \rho(X, next(\Gamma)) + \sum_{i=next(\Gamma)}^{N-1} \rho(G_i, G_{i+1}) \qquad \text{A2-2}$$

The progress distance is the total minimum initial plan distance minus the minimum distance remaining:

$$D_P = R - D_{rem} \qquad \text{A2-3}$$

A distance-normalized version of the progress distance, the progress fraction, is given by:

$$F_P = D_P / R \qquad \text{A2-4}$$

The progress fraction varies from 0 to 1 over the course of a plan execution.

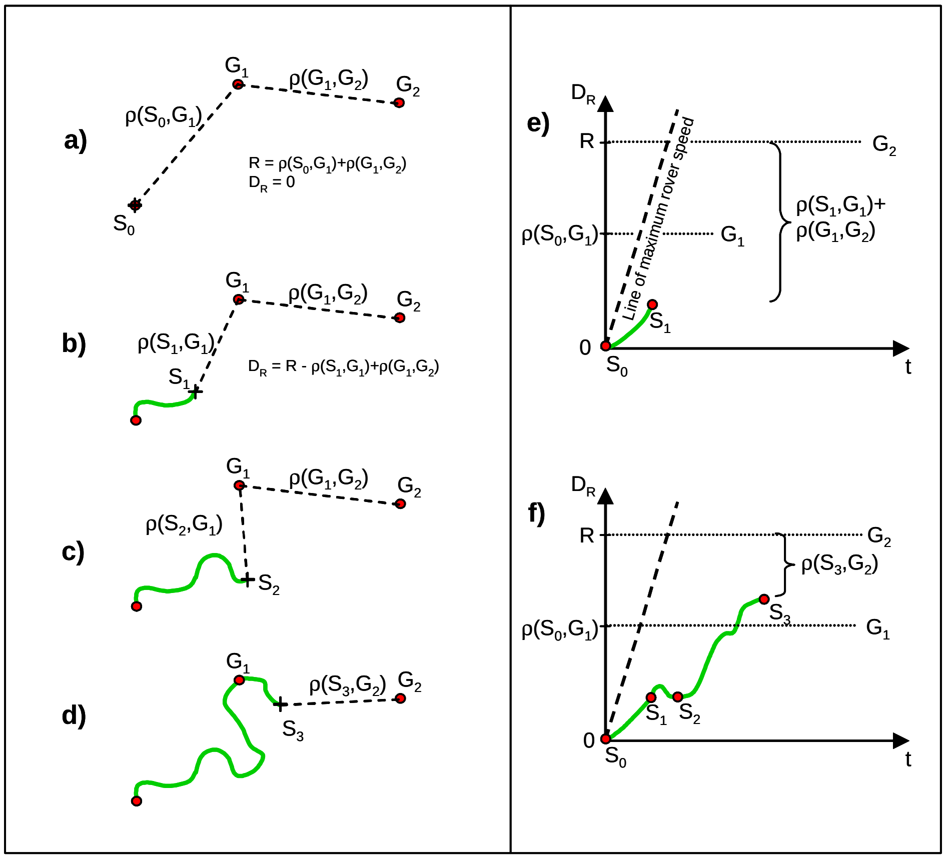


Figure A2-1: Plots of Progress Distance for a Hypothetical Plan

The progress distance is a measure of the progress of the robot along the minimum-distance trajectory through the remaining goals. According to this measure, progress can only be made by reducing the minimum distance remaining. Moving away from the next goal results in negative progress, and moving in a circle about the goal results in no progress. Figure A2-1 illustrates progress distance for a hypothetical spatio-temporal plan. The left side of the figure shows a sequence of four snapshots from a plan execution. The right side shows corresponding plots of progress distance versus time. Figure A2-1a shows the initial start position S_0 and a sequence of two goals G_1 and G_2. The total minimum distance for the plan is shown by the dotted lines. The plot in Figure A2-1e shows the position of S_0 at the axis origin, and the distances to the goals on the vertical axis. In this plot, progress speed is indicated by the slope of a progress distance trajectory. The maximum robot speed is given by the dashed line - no trajectory can advance towards the goals more quickly than this line. In Figure A2-1b, the robot has made progress towards the goal at position S_1. Note the trajectory from S_0 to S_1 in the plot of progress distance in Figure A2-1e. Since the path does not advance directly towards the first goal, the progress trajectory slope is shallow. In Figure A2-1c, the path moves from position S_1 to S_2. In the corresponding progress distance plot in Figure A2-1f, the curve shows no net progress in this move. The reason is that positions S_1 and S_2 are equally distant from the next goal. The final path frame shows the robot at a position S_3, having completed the first goal. The progress curve in Figure A2-1f shows how the turns of the path affect progress over time. Though not shown in the figure, if a robot stops, it produces a constant interval in the progress distance plot.

Progress distance is useful because it is a measure of the directed progress to cover the distance between goals. It reflects the indirection a path might take to avoid obstacles, and also any stationary periods that would otherwise be invisible in a purely spatial plot. Note that progress distance cannot distinguish between stationary activities and mobile activities that stay a fixed distance from the next goal. However, taken together, the path plot and progress distance plot contain sufficient information to determine the spatio-temporal state of the robot. Progress distance plots are used extensively throughout this thesis to illustrate spatio-temporal plan behavior.

Glossary of Terms

Atomic Resource	A resource whose availability can be expressed by a Boolean variable, such that the resource is either available or unavailable. Examples include spacecraft or rover components like cameras, science instruments or motors.
Complete/Completeness	A search algorithm that is guaranteed to find a solution if a solution exists is complete.
Ephemeris	A sequence of values describing the time-varying position of a celestial object in the frame of an observer.
Local Constraint	A constraint whose violation depends only on the current state. The same as an obstacle.
Local Path Planning	Path planning that utilizes only rover-local data, and because of the fine spatial planning resolution, often considers the finite size of the rover, finite turning radii, and may consider vehicle dynamics.
Global Constraint	A constraint whose violation depends on the entire state history.
Global Path Planning	Path planning that utilizes a combination of locally-derived and globally-defined data and, due to planning over long distances at coarse resolution, that generally neglects the finite size of the rover, finite turning radii and vehicle dynamics.
Metric Resource	A resource whose availability can be expressed as a floating point number between 0 and 1, where 1 is available, 0 is unavailable, and numbers between 0 and 1 express partial availability. Examples include energy, solid state memory, and communications bandwidth.
Monotonic Resource	A resource whose level either increases or decreases monotonically, for example charge in a non-rechargeable battery.
Non-Monotonic Resource	A resource whose level varies non-monotonically, for example the battery charge in a solar-powered vehicle.
Obstacle	A state or set of states though which paths cannot pass. Same as a local constraint.
Plan Stability	Determined by the degree to which plans vary in successive re-plans during a mission execution. Stable plans vary predictably in response to evolving initial state, while unstable plans vary erratically and may oscillate in response to small variations in cost maps or initial state.
Progress Distance	See Appendix 2.
Resolution Optimal	Optimal to the resolution imposed by the planning representation.

Sun-Synchronous Navigation	A strategy for planetary polar navigation that synchronizes travel on a path loop with the sun to enable plentiful solar energy, foster benign thermal conditions and permits repeated loops on successive days.