Symbiotic Planning for Planetary Exploration

Joseph L. Amato

CMU-RI-TR-16-09

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

May 1, 2016

Thesis Committee:
William "Red" Whittaker, Co-Chair
David Wettergreen, Co-Chair
Bernardine Dias
Christopher Cunningham

Abstract

Planetary exploration missions avoid the destinations that offer the greatest scientific payout because they entail risks too great for a primary rover. Given the high costs of sending to extraterrestrial bodies, even the small possibility of losing a primary rover precludes the exploration of important, yet risky features. The solution to this problem is a new paradigm for planetary exploration wherein the primary rover is accompanied by multiple cheaper, lighter, and more expendable companion rovers with differing capabilities. These rovers would complement each others strengths and weaknesses through a strategy called Symbiotic Exploration.

Several algorithmic challenges must be solved before symbiotic exploration in planetary environments can become a reality. The less featured companion rovers will often lack a direct line of communication to Earth, requiring that they remain within communication distance of the lander or more capable primary rover to relay data. Similarly, the secondary rovers may sacrifice the large, expensive solar panels and electric generators for a high-density rechargeable battery. Such a configuration would require resource aware planning, as well as occasional rendezvous with a rover or base station capable of recharging or swapping out that battery. A symbiotic planner must be able to develop plans in highly dynamic environments while meeting these constraints.

Distributed Path Consensus (DPC) is an iterative approach to enforcing time-dependent multi-rover rendezvous constraints. It is limited, however, in that it can only apply such constraints based upon the time domain, and the value at which those constraints apply must be identical between the two rovers. This research proposes an evolution of the DPC algorithm (called DPC.TF) that allows it to enforce rendezvous constraints based upon any arbitrary resource, including those that are non-monotonic, even when their values differ across rovers. Furthermore, it allows for complex rendezvous and maximum separation distance constraints to be specified through an intuitive syntax.

A planner based around the DPC.TF algorithm was developed, and its correctness proved on synthetic data. The planner generated routes using actual Lunar data for a variety of multiple rover configurations and symbiotic constraints. This research was able to show that routes do exist to high-interest, permanently shadowed sites while maintaining symbiotic constraints. Furthermore, capabilities that would be required of each rover to explore these sites was analyzed and determined. Such regions have been previously considered inaccessible but, through the paradigm of Symbiotic Exploration, can be thoroughly explored.

Acknowledgments

Foremost, I want to express my gratitude to Red Whittaker and David Wettergreen for their guidance and support throughout my research. I came into CMU not knowing what I wanted or what I could do, and they steered me towards one of the best projects I could have hoped for. Through working with them, I have become a better researcher, writer, and roboticist.

I also thank Chris Cunningham for helping implement the planner used in this work, Heather Jones and Nate Otten for supplying the Lunar data used in this research, and the rest of the Planetary Robotics Lab for helping me stay sane these last two years.

Thank you to all those who helped along the way. To my family: I wouldn't be where I am today without your two and a half decades of loving support. To my old Team Oryx crew from WPI, Jon Anderson, Tom Carlone, Michael Fagan, Taskin Padir, and Ken Stafford: Not only were we a great team, but it was also one of the best preparations for the rigors of graduate school at CMU that I could have had. Finally, thank you to Elizabeth for encouraging me to stick with it, even when times were tough. I couldn't have done it without you.

This work was supported by a NASA Space Technology Research Fellowship, as well as a National Robotics Initiative grant from NASA.

Contents

Chapter 1

Introduction

1.1 Motivation

The history of robotic space travel is one of solitary pioneers, operating alone in their task of planetary exploration. There are some exceptions to this rule; the Pioneer 13 mission consisted of multiple probes launched to different regions of Venus, and Sojourner was accompanied by the static base station Pathfinder during its mission to Mars. But by and large nearly every rover sent to explore extraterrestrial bodies, from the Lunokhod on the Moon, to Curiosity on Mars, involved a single rover as the primary mission asset.

The popularity behind the single rover approach is clear: planetary exploration is an expensive and complicated feat. Designing, manufacturing, assembling, and testing space-grade rovers is a costly and time consuming project. Curiosity spent nearly two years in its design phase and another two years for assembly and testing, at a cost of nearly $2.5 billion. Launch costs, which can be as high as $1.6 million per kilogram sent to the Moon, only add to the expenses.

These *astronomically* high costs have led to a justifiably risk averse attitude to exploration missions. Yet, it is the risky planetary regions that offer the greatest scientific payout. Craters, caves, and canyons show promise for harboring volatiles like water and hydrogen [1]. Pits on the Moon are safe havens from radiation, micrometeorites, and temperature variations and show potential as permanent exploration habitats [2]. The rugged terrain and precipitous slopes of all these features are too dangerous to risk primary mission assets. Given the high costs of sending rovers to extraterrestrial bodies, even the small possibility of losing the singular mission rover precludes the exploration of important but risky features.

Even within the regions where rovers can safely operate there exist challenges. Most rovers are designed to minimize operations while outside of communications to Earth, so that recovery is possible if something goes wrong. This is particularly challenging in lunar missions, where each lunar cycle takes more than 27 days. Furthermore, the wild temperature swings on the Moon make operating in shadowed regions dangerous, as components are prone to freezing. Because of these requirements, there are fixed time windows when the rover can operate. Having only a single rover significantly limits the amount of exploration and scientific work that can be conducted during these time windows.

But risk to the rover is not the only challenge that must be factored in when dealing with the single rover architecture. Some missions must account for the risk of damaging or contaminating pristine regions. Recently, the Mars Reconnaissance Orbiter found that the recurring slope lineae (RSL) scattered across the walls of Mars's craters are caused by briny water that is currently flowing. These RSL, NASA believes, offer the best possibility for discovering Martian life. While the potential scientific payout of investigating these regions is massive, they are considered off-limits by ground rovers due to the risk of contaminating them with Earth-based life.

In prior missions, the single-rover architecture proved to be an ideal configuration, balancing the challenges of cost, complexity, risk, and capability effectively and efficiently. While such missions carried with them a great deal of risk, future missions will take rovers to even more hazardous locations in search for even greater reward. In such instances, a single-rover architecture is no longer the ideal approach. Instead, a new paradigm for planetary missions must be explored. This research proposes such an approach in the form of Symbiotic Exploration, where multiple heterogeneous rovers operate in a symbiotic fashion, complementing each other's strengths and weaknesses and leveraging the differing capabilities of the rovers. Through this approach, the aforementioned challenges become tractable, enabling far greater exploration capabilities than achievable with a single rover.

1.2 Symbiotic Exploration

Symbiotic Exploration is the idea that multiple complementary robots will be far more effective at exploring the hazardous, yet most valuable regions of planetary bodies without introducing extraneous risk on the mission. Rather than send up a single rover to accomplish the mission's tasks, that rover will be accompanied by a number of smaller low-cost rovers. These secondary, or companion, rovers would, by virtue of their number and lower cost, be more expendable than the primary rover, and therefore able to take greater risks.

The fully featured primary rover can operate in known safe areas while using the secondary rover to explore dangerous, yet scientifically valuable areas. This could include scouting out the safest route to a region of interest before sending the primary rover there. The secondary rovers could also be useful in identifying which prospective regions either contain or lack volatiles. Rather than send the main rover down to a dangerous area that may not have any scientific value, a symbiotic architecture would ensure that all risks the primary rover must take are justified.

While symbiotic configurations are popular in search and rescue applications, their application to planetary exploration has been less thoroughly explored. The closest example of its usage on a mission was on Pathfinder, consisting of the wheeled rover Sojourner and an instrumented, but immobile, lander. Recently, there has been a renewed interest in heterogeneous configurations for use on planetary missions. Jet Propulsion Laboratory's Axel Rover tethers itself to another rover in what is known as a marsupial arrangement, allowing it to safely traverse down precipitous slopes [3] and to combine with another Axel Rover and a central module to form a four-wheeled Du-Axel rover [4]. The Tokyo Institute of Technology takes another novel approach to symbiotic rovers with their SMC rover in which each wheel of the SMC rover is itself a detachable rover capable of movement and manipulation [5]. While these developments are promising, they have focused primarily upon the hardware challenges of the symbiotic architecture.

There has not been nearly as much research into how to address the algorithmic challenges of symbiotic planning for planetary exploration, of which there are many. Limited resources necessitate plans be carefully constrained to limit their expenditure. Indeed, it is possible that the companion rovers, in an effort to reduce costs, will forgo expensive thermoelectric generators and solar panels in favor of a cheaper rechargeable battery. Such a compromise would no longer allow them to be entirely self-reliant. Instead, they must occasionally rendezvous with a more capable rover to recharge their energy stores or swap out their batteries. Similarly, a direct line of communication to Earth, requiring a large and powerful antenna, may be replaced with a smaller module suitable for short-range (< 10km) communications. In such instances, they would rely upon the primary rover to act as a relay to Earth, and must plan their paths so as to remain within range of each other. These are the challenges that must be addressed if the paradigm of Symbiotic Exploration is to be realized.

1.3 Contributions

The key contribution of this work is a planning algorithm that is capable of generating routes for multiple heterogeneous rovers that address the constraints inherent in symbiotic exploration. These symbiotic constraints include resource-aware planning, allowing for rendezvous based off of any arbitrary monotonic or non-monotonic resource, and guaranteeing that the rovers stay within a specified distance for the duration of their trajectories.

This algorithm has been implemented in software, and the resulting planner is validated on synthetic data to prove its performance. Plans are generated using actual data from the highly dynamic regions of the Moon that future missions will explore. These plans show that viable routes with the given symbiotic constraints do exist and that Symbiotic Exploration allows regions of the Moon to be explored that would otherwise be considered off-limits. Finally, an analysis of different rover capabilities and their effects on the resulting routes is explored, so that the most effective configuration for future missions can be chosen. The goal of this research is not to implement a fully mission-capable algorithm and software package, but to serve as the foundation for future work into the problem of symbiotic planning.

Chapter 2

Background

This chapter discusses past path planning research that is relevant to symbiotic exploration. The scope spans from the most commonly used planning algorithms to approaches that allow for rapid trajectory planning for multiple robots in high-dimensional spaces. Finally, existing work that relates specifically to symbiotic constraints is reviewed and the benefits and drawbacks to each of those works is analyzed.

2.1 Rover Motion Planning

Before discussing planning for multiple robots, it is necessary to have a solid understanding of the fundamentals of single rover planning. This section first discusses common methods for generating deterministic plans on search graphs. Next, it delves into stochastic methods that are suitable when searching the entire graph deterministically is not tractable.

2.1.1 Graph-Based Planning

Graph based planning is one of the oldest domains within path planning, with some of the earliest methods of graph-based search (such as depth-first search) dating back to the 19th century [6]. While algorithms such as depth-first search and breadth-first search have been around for a long time, they have many pitfalls (such as incompleteness or no guarantee of optimality) that lead them to be widely unused in modern day planning.

Dijkstra's algorithm (also known as uniform-cost search) was an enhancement to existing methods that was both optimal and complete, that is, it will always find a solution and that solution is guaranteed to be the shortest possible path [7]. Dijkstra's algorithm finds the shortest path between two nodes on a graph through the use of a priority queue that returns the lowest cost node expanded. When a node is expanded, it is added to the queue with a cost equal to the cost of the edge between the two nodes plus the total cost of all edges between the starting location and the current node. Thus the search will radiate outward along the shortest possible paths. While Dijkstra's algorithm is both complete and optimal, it is slow to compute. Because Dijkstra's algorithm is uninformed, (ie. it makes no assumptions about its environment), it will often expand many more nodes than are necessary to find an optimal solution.

If information is known about the graph prior to planning, heuristics can be used to significantly speed up the runtime performance of the planner without sacrificing optimality [8]. This is the approach taken by the widely-used A* planning algorithm [9]. In A*, a heuristic acts as an estimate of the final cost to reach the goal from any node. Planning is performed similar to Dijkstra's algorithm, with the exception that the cost of a node is equal to the total cost to get to that node from the starting point, plus the estimated cost to reach the goal from this node. The result is that

node expansion will be drawn towards the direction of the goal, rather than expand out uniformly from the starting location. The only precondition to using a heuristic is that it must be admissable (ie. it must always underestimate or perfectly estimate the cost to the goal), otherwise optimality of the solution is not guaranteed.

While the A* algorithm is over half a century old, there have still be many modern improvements to it. Phillips et. al proposed parallelizing the neighbor state expansion stage of the A* algorithm [10]. This approach is ideal for situations where neighbor expansion is costly, as will likely be the case in the symbiotic [planning due to the calculation of multiple constraint costs. Recently, the use of multiple heuristics has become a popular method of improving the performance of A* [11]. With the multi-heuristic approach, a number of inadmissable heuristics, alongside a single consistent heuristic are used for planning. Each heuristic is given its own priority queue. If a better path to a state expanded by any of the heuristics is found, that information is updated across all of the priority queues. In situations where the heuristic may vary from task-to-task or when the search space is highly dimensional, the use of multiple heuristics can significantly speed up search time while guaranteeing suboptimality bounds[12]. Research has also explored improving the performance of multi-heuristic A* on highly dimensional problems even further by dynamically generating heuristics[13].

When dealing with problems operating over many dimensions, particularly when the search space changes within those dimensions, simply applying multiple heuristics is not enough to improve planner performance. Such a problem is typical in environments with dynamic obstacles that require planning within the time domain, or problems where there are multiple variables that are attempting to be minimized. Such a scenario is likely to occur in planetary environments, where lighting changes frequently and paths will likely be optimized to reduce time, distance traveled, and energy expenditure simultaneously. In such scenarios, state dominance functions are required to prune nodes from expansion [14]. A state dominance function operates during the neighbor expansion step of search. If the state that is about to be added to the search queue is strictly worse than either another state in the queue, or one that has already been expanded, it is discarded. This prevents the search space from growing exponentially, and puts an upper bound on the number of possible states that can be expanded for a given node. Choosing a proper state dominance function will be essentially to ensuring that the search space within the symbiotic scenario remains tractable.

While the above algorithms are ideal when operating in known environments, having a complete map of the search space beforehand is not always feasible. Anthony Stentz developed a graph based planner optimized for fast replanning in partially unknown environments, known as D* (short for Dynamic A*) [15]. D* differs from A* in that it plans from the goal to the robots current location, maintaining a cost to the goal and a backpointer to the next node to the goal for each node in the graph. When the rover encounters an unforseen obstacle, previously visited nodes have their costs raised or lowered according to the new information. The benefit to this approach is that the plan does not have to recalculate the costs for all nodes in the search space when it comes across a new obstacle, only the nodes that are directly affected by that obstacle. Since D*'s initial publication, there have been many variants that have improved upon it, including Focussed D* [16], D* Lite [17], and Field D* [18].

Other methods of improving the search space have focused less on the planning itself and more on how the search graph is represented. One popular approach is to decompose the graph into hierarchical structures such as quad-trees [19]. Often times, much of the search graph is either empty or will be unsearched. In such situations, there is no need to finely represent the entire search space. With quad-tree decomposition, the search graphs are initially represented by large nodes. The more interesting nodes are broken down into much smaller nodes, while sparse regions remain large [20]. This allows for fast traversal through large portions of the search space without sacrificing fidelity where needed.

2.1.2 Stochastic Planning

Sometimes, deterministic methods of planning are not enough to reduce the planning time to an acceptable level, even with the optimizations mentioned above; the search space is simply too large. Such a scenario is particularly common in high-dimensional problems and dynamic environments. In such instances, stochastic methods present a strong tradeoff between optimality and runtime performance.

Probabilistic roadmaps (PRMs) are one commonly used method for quickly generating tractable search spaces from intractable environments [21]. The PRM algorithm (shown in in Algorithm 1) operates by randomly sampling points from the free configuration space. As it samples more and more points, it looks in a neighborhood region. For each of those neighbors, an edge is generated if none exists and a valid edge can be formed between the two. This process continues until a valid path between the start and goal can be found. While the sampling approach used by PRMs is very fast, it suffers from a number of problems. First, it is difficult to constrain the edges between nodes to a specific kinematic model. Second, it is only asymptotically complete. Although a valid solution may eventually be found, it may end up taking longer than other deterministic methods. This challenge is compounded by the fact that the random sampling conducted by the PRM algorithm may leave large areas of the search graph that could contain a valid solution unexplored.

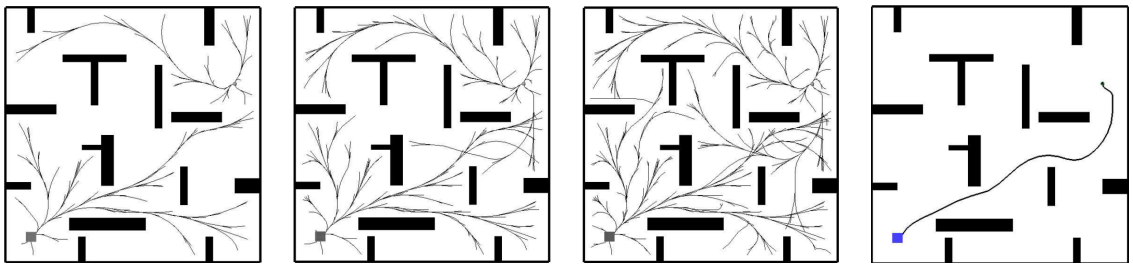
Algorithm 1 PRM Construction [22]

```
V ← ∅
E ← ∅
loop
    c ← a randomly chosen node from the free configuration space
    V ← V ∪ {c}
    N_c ←a set of neighbors of c chosen from V
    for all n ∈ N_c do
        if (c, n) ∉ E & L(c, n) ⊂ C_f then
            E ← E ∪ {(c, n)}
        end if
    end for
end loop
```

Where:
V: The set of all nodes in roadmap
E: The set of all edges between two nodes
C_f: The free configuration space
L: A local planner that generates a path between two nodes.

Rapidly-exploring random trees (RRT) are an approach to addressing many of the problems with PRMs. The core idea behind RRTs is to use a space-filling tree that expands from the starting configuration towards large unsearched regions through branches that meet the robot's kinematic constraints [23]. Figure 2.1 shows an example RRT exploration at different stages of the search, as well as the final path for a nonholonomic robot. In this example, RTT expansion occurred simultaneously from the starting point of the robot and the goal location.

Note that, while RRTs find a solution quickly with few node expansions, there is no guarantee that the path be even close to an optimal solution, only that it will be a valid one. Karaman proposed a variant of RRTs (known as RRT*) that attempted to address this problem by being asymptotically optimal [24]. Rather than simply extending nodes only from branches like the core RRT algorithm, RRT* would take the additional step of searching for all nodes within a neighboring region to the newly expanded node and add edges between the two. If a better path between nodes was found, weaker edges would be removed. Additionally, RRT* would continue searching and

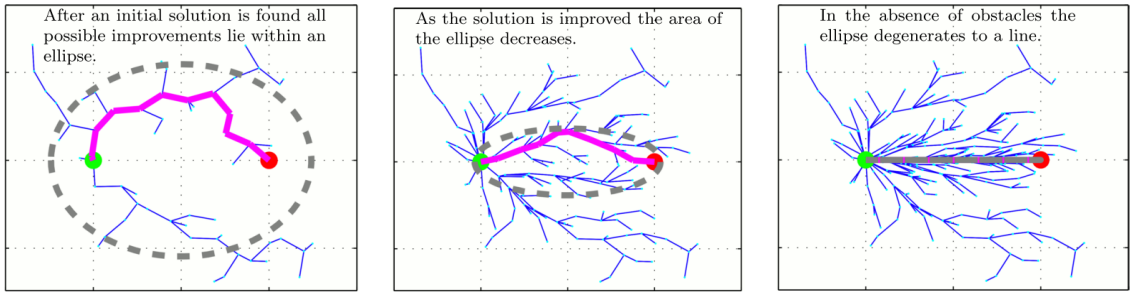


(a) 500 Nodes Expanded (b) 1000 Nodes Expanded (c) Final (1582 Nodes) (d) Final Path

Figure 2.1: RRT exploration at various stages of planning [23].

improving its plan, even after a solution toward the goal had been found. This gives RRT* the useful quality of being an anytime algorithm where an optimal solution can be found if given enough time, but the algorithm can be terminated earlier if time constrained while still returning a valid solution.

If the purpose of the planner is to minimize path length, Informed RRT* is one approach that can be used to significantly speed up the process of finding an optimal solution. Informed RRT* works in a manner similar to RRT* with the exception that, once a path between the start and goal nodes is found, the search space is pruned significantly to allow faster convergence upon the optimal solution [25]. The search space is pruned by constructing a multi-dimensional ellipsoid with both the start and goal nodes treated as focal points. The width of the ellipse is equal to the total length of the shortest found path between the two points, while the height is equal to $\sqrt{d_{best}^2 - d_{min}^2}$. Only nodes lying within this ellipse are sampled. As better and better paths to the goals are discovered, the ellipse shrinks until an optimal solution is found. Figure 2.2 shows the process of Informed RRT*s ellipsoid pruning.



(a) 58 iterations, $d_{best} = 148.24$ (b) 175 iterations, $d_{best} = 107.12$ (c) 1142 iterations, $d_{best} = 100$

Figure 2.2: Informed RRT* converging towards an optimal solution. The start and goal states are represented as green and red dots, respectively, the current best solution in magenta, and the sampling ellipsoid in grey. [25].

There are challenges to using Informed RRT* in the symbiotic scenario, however. First, the presence of additional symbiotic constraints that inherently pull the rover away from otherwise ideal paths may not be compatible with Informed RRT*s method of pruning the search space. Similarly, Informed RRT* only works for paths optimizing monotonic qualities, such as distance or time. If trying to find the path with the lowest energy cost, their method of pruning may not be applicable.

2.2 Multi-Agent Planning

There are many advantages to multi-agent configurations over those with only one rover. Multi-robot systems can often complete tasks faster than with a single rover, explore greater area faster, and offer redundancy in case of failure [26]. Yet, just as there is a material cost in adding more robots, there is an algorithmic cost as well [27].

Planning for multiple robots increases the complexity significantly. Besides the problem of simply having more paths that require planning, additional robots often add newer, more challenging constraints. One of the core constraints among these is collision avoidance. Planning for multiple robots creates a dynamic challenge in otherwise static worlds, as the location of the other robots varies over time [28]. This added dimensionality can quickly cause the planning problem to become intractable, as common heuristics lose their consistency guarantees. Indeed, even when the path is fixed and all that is being determined is the velocity profile simple, the problem can be too difficult to solve with simple algorithms [29].

Simeon et al. proposed a novel method for decomposing movements of multiple robots along fixed paths by modeling the problem using a coordination diagram that represented where along each robots path a collision could occur by decomposing possible collisions into a number of elementary cases [30]. This allowed them to quickly compute and explore the collision graph without having to compute the exact shape of any obstacles in the map, or the exact configuration that causes a collision. Their method proved effective for planning problems involving as many as 150 robots.

When not just the motion profiles, but the paths themselves must also be found, the challenge of the problem increases even more. A simple method of planning in the time domain is to simply treat time as an additional dimension and run A* search over the three dimensional (x, y, t) graph [31]. As mentioned before, however, the search space will quickly overwhelm such an algorithm. Wang and Goh proposed a spatio-temporal variant of A* to plan paths for multiple robots while avoiding collisions that sought to address the dimensionality problem [32]. Their approach was to perform A* search on the normal two-dimensional graph, while maintaining a temporal occupancy table. If, while evaluating whether or not a state should be expanded, they find that it will result in a collision, they will perform a backwards search to find the closest antecedent state that the robot can safely wait in until the collision would no longer occur. Their approach proved to be several orders of magnitude faster than three-dimensional A*, but still suffered performance problems when operating over large graphs and timespans.

Svestkas and Overmars approached the challenge of high dimensionality through stochastic methods [33]. First, they generated a probabilistic roadmap through the same methods discussed in Section 2.1.2. This roadmap is then decomposed into a number of subgraphs containing nodes that are considered close together. They then make the assumption that a robot only needs to plan taking into account the actions of another robot iff a robot is operating in the same subgraph. Otherwise, it can generate that portion of the plan locally without regards to the location of the other robot. Their approach has the advantage that the roadmap and subgraph construction can be done through a preprocessing stage and cached for future usages.

Another commonly researched area in multi-agent planning is how best to explore an unknown environment. The strategies used to maximize exploration gain vary significantly. Thrun et al. discuss an exploration and map builing strategy for normal rovers that allows them to localize without odometry through scan-matching [34]. Robots will periodically travel through already mapped regions in order to relocalize themselves if lost. This approach sacrifices maximum exploration in favor of accurate localization.

More coordinated approaches to multi-robot planning and exploration are also popular. Burgard et al. combine the domains of planning and task allocation in their multi-robot strategy. They describe a utility function that weighs the benefits of exploring a given target region with the cost associated with getting there [35]. Using a known sensor model, they predict an expected visibility

range from when entering an unexplored room; the larger the expected visibility, the higher the value. They then calculate the cost to get to that region using value iteration. In later research, they also discussed how their method could be used to enforce communications restraints on robots [36], a topic which will be discussed in greater detail in Section 2.3.3.

Although many of the above methods were designed for homogeneous teams of robots, they can be adapted for use with the heterogeneous architecture used in symbiotic exploration. Burgard's approach, for instance, could be adapted to use a heterogeneous team by changing how the penalty function during value iteration is applied. Still, it may be better to use an approach designed for handling robots with varying capabilities. Buehler describes a framework for tackling planning and task allocation problems for diverse robot teams [37]. Given an arbitrary set of actions, states, and preconditions, they designed a generic algorithm that produce optimal plans for robots. Their key point was that the framework they developed was truly platform independent and could be used for robot teams with wildly varying capability just by changing the preconditions and actions within the planner.

The concept of "marsupial robots" is another commonly proposed architecture for heterogeneous robot teams, particularly in the fields of search and rescue, though can be equally suitable for planetary exploration. In the marsupial architecture, a large robot carries a number of distinctive robots to a launching point, and releases them when it can go no further. Each of these smaller robots may have a specific task it is most capable at accomplishing. Murphy's work into shape-shifting robot teams for urban search and rescue touches upon a number of aspects of symbiotic exploration[38]. The approach she describes is a highly symbiotic one, where the cooperation of bother the "mother" and "daughter" rovers amount to something greater than the sum of their individual contributions [39]. An example that she discusses is the ability to offload computing resources from the daughter robot to the mother robot, allowing the daughter to be cheaper, more light weight, and more specialized. She also discusses the ability to use the mother robot as a sort of communications relay for the daughter robot, a topic of great interest in symbiotic planning.

Wurm et al. also studied how best to leverage marsupial robot teams for exploration and planning. Their approach focused on many of the temporal problems of the marsupial architecture. They argued that planning for marsupial teams requires is an inherently temporal problem, while many of the actions can be done in parallel, but some actions such as (deploying, collecting, and recharging) have explicitly time costs that must also be considered [40]. Furthermore, just getting to a goal, or meeting up with another robot is not always enough. Sometimes, the robot must remain within a region long enough to carry out its task. Their approach was to treat the acts of deploying and conducting experiments at the goal as a part of the plan. Thus, if you could reach the goal in time, but would not be able to conduct any of the designated experiments once that goal had been reached, it would be considered an invalid plan. In the case of symbiotic planetary exploration, this requirement is particularly necessary. Recharging often takes a great deal of time and would require the robots to stay together. Similarly, given the time varying planetary landscape, a situation may arise where it is not ideal to arrive at a goal in the shortest possible time if a shadow will cause the robot to have to leave the area before it can complete it's experiments.

2.3 Symbiotic Planning

Symbiotic planning for planetary exploration requires additional constraints beyond the traditional multi-agent planning and exploration algorithms described above. The hazardous landscape with time varying communications and lighting requires that any planner take careful measure of its resources. If a rover is unable to replenish lost resources or relay communications on its own, it must occasionally rendezvous with those capable of doing so. Finally, they high level of cooperation inherent to symbiotic exploration necessitates constant communication between the rovers. Any planner designed to operate in the symbiotic scenario must take into account these symbiotic

constraints.

2.3.1 Resource Constrained Planning

One of the largest constraints in a symbiotic architecture stems from resource availability and replenishment. Given the differing capabilities and approaches of the symbiotic rovers, it is possible that one of the rovers is limited in a specific resource, such as power or direct-to-Earth communications. The plans that will be generated will need to take careful measure not to exceed resource limitations, particularly with regards to battery life of rovers. This requires that any planner used for symbiotic exploration must incorporate models for resource consumption and generation.

Integrating resource costs into a planner is challenging because cost functions for power are neither monotonic nor consistent. Such challenges were investigated by Tompkins in his PhD thesis work [41]. Tompkins developed TEMPEST: an approach to path planning that optimizes path cost based upon a multi-dimensional set of available resource. TEMPEST is a generalization of the D* algorithm that factors in the availability of non-renewable monotonic resources (such as non-rechargeable battery energy or finite-lifetime components) and replenishable resources (such as conventional battery energy, available memory, or communications bandwidth) for mission-based path planning. By simultaneously constructing the plan in the spatial, temporal, and energy domains, he was able to develop globally optimal plans that satisfied high-dimensional constraints such as energy expenditure.

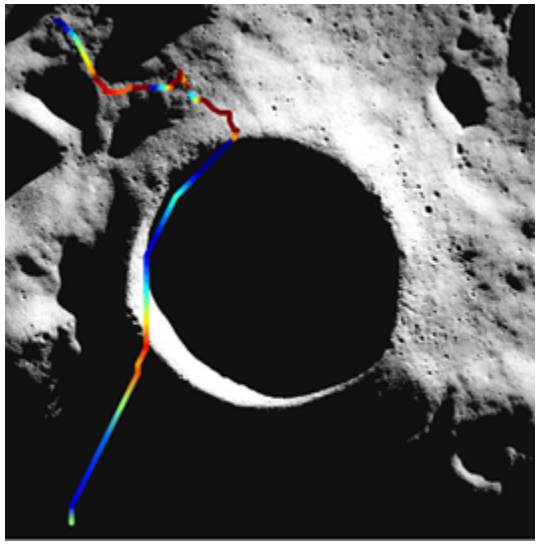


Figure 2.3: Simulated traverse around Shackleton Crater generated using energy and temperature constraints. The path is colored according to the amount of energy the rover has remaining, where blue and red indicate minimum and maximum energy levels, respectively.[42]

Recently, Whittaker and CMUs Planetary Robotics Lab has also had an increased focus in resource constrained planning, having developed an A*-based hierarchical planner that considered energy and thermal constraints with physics-based models for resource generation and depletion [42]. Their planner enabled a rover to gain or lose energy by having multiple discrete edges between each location that corresponded to different changes in energy (shown in Figure 2.3). This planner was optimized for rapid replanning where the shadows vary significantly with time or location. Of particular note are optimizations they used to improve the performance of the planner. As noted in section 2.1.1, planning in the time dimension increases the number of possible states exponentially.

Cunningham addressed this problem through a state-dominance function that significantly limited the number of states that required exploration.

Otten has taken a similar approach to Cunningham to address the problem of power depletion for rovers by always travelling through sunlit regions of the Moon using a strategy called Nomadic Exploration[43]. In the Nomadic Exploration scenario, a rover would continuously circle a crater or region of interest while minimizing the time spent in shadow and out of communication with Earth, multiplying mission duration by orders of magnitude. He was able to conduct search over extended periods of time by pruning the search space through connected-component analysis. His approach implemented a fixed-length time window in which a rover was allowed to travel through shadowed areas. By setting this value to zero, the rover would avoid unlit regions entirely. Additionally, a no-motion constraint was applied when no communication link was available to Earth.

Wu and Ju likewise explored planning while taking into account communications constraints when they developed their mission-oriented planner for planetary exploration [44]. Their approach treated the task of planning as a sort of state machine, where each state included both the state of the robot(such as remaining battery life), and the state of the world (such as communications or lighting shadow). Given the configuration of a state, the rover could take one of any number of tasks (moving, transmitting data, etc.) that acted as a transition to a new state. The tasks available to take were limited by the state. For instance, a robot could not transmit data or drive if it lacked a direct line of communication to Earth. This state machine was then used in conjunction with an A^* planner to produce a series of actions to take, rather than points to travel to. One drawback of their approach is that it did not take into account the possibility of taking accomplishing multiple tasks at the same time, such as transmitting data while driving.

One drawback that all the above methods suffer from is that they only operate under the assumption that the mission is conducted with a single rover. Many of the above planners are designed to optimize for a single set of capabilities, for instance, the rover must stay out of shadows, or can only travel over certain terrain. The symbiotic scenario presents a challenge for these planners, as each robot in the team can have very different capabilities.

2.3.2 Multi-Agent Rendezvous

While minimizing each individual rovers' resource expenditure is an essential task, the symbiotic effects of a multi-rover architecture must be factored in to planning. Indeed, the symbiotic architecture introduces many additional actions that can affect resource constraints. For instance, it is entirely possible that a rover may drop off samples with another rover if its collection mechanism is full, or recharge its battery if running low. The latter example is particularly applicable to planetary exploration; it may not always be the case that a rover can operate completely self-charged. In the case of small, cheap rovers which do not have large solar arrays are expensive RTGs, they must relying upon an external power source for recharging. This requires the robots to occasionally rendezvous based on resource constraints if necessary.

Path planning for coordinated arrival at predetermined locations was explored by McLain and Beard in the context of UAVs avoiding radar installations and arriving at targets simultaneously [45]. Their approach modeled the search graph as a Voronoi diagram to maintain maximum distance from "threat locations" and chose which edges to traverse based upon the total path length and a risk factor assigned to each edge. The paths were then perturbed slightly, so that they would all be of equal length to the longest-distance path to guarantee synchronized arrival at the goal locations. Although McLain and Beard used separate locations for the UAVs to simultaneously arrive at, the desired outcome could be achieved by having the target locations be identical across all rovers. A major drawback of their approach requires that the rendezvous locations be known beforehand. Given the complexity of planetary landscape, it may not be possible to analytically predetermine optimal rendezvous points before path generation.

A similar task was undertaken by Mathew, Smith, and Waslander, but instead of having the

UAVs arrive at goal locations at the same time, they would dock with mobile robots to recharge before continuing surveillance tasks[46]. In their scenario, each UAV has a predetermined path that it may follow and selected discrete portions of that path to be marked as charging segments, where a rendezvous is must to occur to recharge the UAV. Once the charging segments where assigned, they modeled the problem as a multi-agent TSP, where each charging point must be visited at least once by any single charger robot. Though they were able to solve the problem for complex paths involving numerous rovers, it suffers from the need to know at least one of the paths beforehand, as well as know all of the charging points prior to planning.

Meghjani and Dudek took an online approach to the rendezvous problem that allowed them to maximize exploration area that a team of robots can cover [47][48]. Their approach was to take a prior map of the are and "thin" it to a skeleton structure using Hilitch's algorithm[49], significantly reducing the search space. They then assigned pairs of rovers with a soft constraint to rendezvous with eachother at predetermined times. As their scenario disallowed communications between rovers, rendezvous points were determined by each rover individually based upon a set of strategies that sought to maximize the probability of finding the other robot at the rendezvous point. Although there method did allow for fast computation of plans, it had many drawbacks. First, because the rendezvous locations were not predetermined, there is no guarantee that the rendezvous would actually occur. It is possible that both robots to not choose the same "meetup" location. In such instances, the robots continue on their exploration and try to rendezvous at another time. Second, while their skeletonization of the graph worked well for reducing the search space, such a method would be difficult to adapt to a time varying environment.

Litus, Vaughan, and Zebrowski explored the problem of determining the optimal rendezvous points between "feeder" robots that need recharging and "tanker" robots that can recharge the workers[50][51]. Their approach was to first combinatorial find what order a given tanker should attend to the worker robots. Once they had determined what order to travel to, they solved the challenge of locating rendezvous points by modeling it as a Fermat-Torricelli problem wherein they minimized the total energy spent on locomotion across all agents [52]. Though their approach is suitable for small search spaces, it is unfeasible in a large, open environment. Furthermore, the algorithm only determines the optimal paths to a meetup location based off of the current position of each rover. It is unable to design a path such that the rovers can travel to a waypoint and meetup sometime in the future.

Distributed Path Consensus (DPC) is one promising algorithm that takes an iterative approach

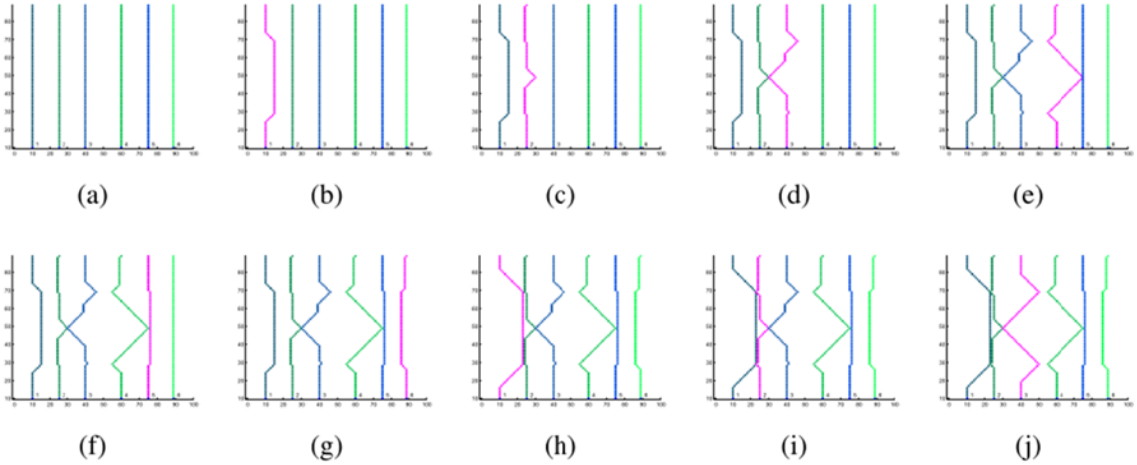


Figure 2.4: Step-by-step run-through of the DPC algorithm for multiple robots[53].

to determining ideal rendezvous points for multiple rovers[53][54]. The key idea in DPC is to constrain the distance between agents by applying a distance-weighted penalty to paths that exceed a location-based distance requirement. DPC iteratively recalculates each agents path while increasing penalties. In this way all paths converge to a minimal cost, although not necessarily the global minima, while satisfying the distance constraints.

Figure 2.4 shows an example of the paths that DPC generates for multiple robots with rendezvous constraints.The pink path in each subfigure indicates the robot that is having its path planned for that iteration. Subfigure 2.4a shows the free path of each robot (starting at the bottom of the graph) to its goal directly above it. DPC recalculates the path for each robot, causing it to edge closer to its partner at a point in time. The robot whose path is being recalculated in the given iteration is shown in pink. The final converged plan for each robot is shown in Subfigure 2.4j.

One of the primary challenges with using DPC in a symbiotic architecture is that it only enforces constraints based on time, and the time when the constraint is applied must be identical between rovers. Even if one were to use resources expended as the constraint determinant, if the resource is expended at different rates between the rovers, the robots will be drawn to the same location but arrive at different times. Having the two robots meet up when only one of them has expended a certain amount of energy is not possible with basic DPC. Additionally, specifying a maximum distance constraint that is active through the entire path (like one used for communication) requires the constraint to be enumerated at every possible timestep instead of capturing it in a single constraint.

2.3.3 Maximum Separation Constraints

Inter-rover communication imposes an additional constraint in the symbiotic rover architecture. The high level of cooperation between the rovers necessitates reliable communication. Such a constraint is further necessary if a dedicated rover-to-Earth communication system is not feasible for one of the rovers, forcing it to rely on other rovers for communication. In order to do so, it must remain with communication distance of its partner rover. This constraint can be applied by enforcing a maximum separation distance constraint between the rovers, such that they remain within communication range of each other for the duration of their trajectories.

Distance constraints between multiple robots during motion planning are commonly applied to force robots to maintain a desired formation during path traversal[55][56]. In these situations, the path for each rover is generally predetermined and the planner seeks only to determine optimal velocity and acceleration profiles to maintain the distance constraints [57]. A much more challenging problem is to account for the distance constraint when developing the multi-rover path itself.

If the robots begin in a connected configuration, a simple method of accounting for the distance constraint is to develop a utility function that ignores movements that will cause a robot to leave the communication network. This is the method used by Rooker and Birk for collaborative exploration of unknown environments [58]. They described an approach that randomly sampled a number of possible movements across all robots and selected the best movement configuration that brought the robots closer to unexplored territory while still maintaining a communication network. This method is best suited for determining paths that maximize exploration area rather than have the robots explore a specific goal.

Potential fields have long been a method of enforcing constraints in multi-robot systems[59]. Pereira, Das, Kumar, and Campos used artificial potential fields to allow robots to travel to individual goals while remaining within communication distance [60]. In their approach, the potential of each cell is based upon the operating mode of the robot and their location relative to the closest robot in their group. If the distance to the closest robot is greater than the desired threshold, the potential of each field is updated such that the two robots will be drawn together while still navigating slightly towards their goal. Once the robots return to communication range, the effect of the distance constraint is set to zero until robots once again drift apart. Though they were able

to show that their method would find a solution that followed the constraints if it existed, the use of potential fields for path determination did not guarantee that the chosen paths would be efficient.

Mosteo et al took a similar approach to Pereira in their attempt to address the problem of multi-robot routing with communications constraints while adding in the challenge of allowing the robots to form communication meshes[61]. These meshes would allow robots to communicate with other robots, even if there was not a direct connection between the two communications robots. Their approach was to treat the search space as a sprint-damper mesh, where communications constraints between robots were treated as "virtual springs" that applied force when the distance to the nearest robot became too large. Goals and obstacles were likewise given a "spring force", with the goals force being a determined according to an arbitrary cost function (based on time, energy expenditure, or distance traveled), and obstacles having a repulsive force according to their risk. The path taken would be determined by summing the forces. Of particular note was the ability of their approach to adapt to run-time constraints by adjusting the forces of the virtual springs if the environment is not as expected. This allows for it to adapt to additional challenges useful to in the symbiotic scenario, such as degraded communications signal strength to to environmental effects or losing line-of-sight.

Mosteo's approach was not without drawbacks. One of the assumptions they made in order to optimize their approach was that all robots were homogeneous and that constraints were identical across all rovers. In the symbiotic scenario this research proposes, such a configuration is not acceptable. The primary and secondary rovers will likely have vastly different capabilities. Furthermore, none of the above approaches take into account the aforementioned symbiotic challenges of resource constrained planning and rendezvous. In order for a planner to be suitable in the symbiotic scenario, it must be able to simultaneously address all of the above constraints when operating on the large dynamic environment that the planetary landscapes require.

2.4 Summary

Overall, there has been a wealth of existing research into each of the different aspects of symbiotic planning, but little has been done to tie all of them together. Traditional multi-rover planning has been heavily focused on maximizing exploration area or aiding in search and rescue missions. Often times these rovers are homogeneous. The Marsupial Rover architecture is one instance is an exception where rovers are meant to complement each other through different capabilities in a manner similar to what is desired with Symbiotic Exploration. Indeed, Symbiotic Exploration is a subset of the Marsupial Rover architecture with specific types constraints on resources and communication.

Most existing resource based planning algorithms have involved rovers acting alone without exploring how a companion rover can affect the plan, particularly with regards to multi-rover rendezvous. Those that have involved rendezvous are either unable to account for variables other than time for determining rendezvous (DPC) or cannot generate plans to goals while tanking into account rendezvous (as in the Frugal-Feeding Problem). A symbiotic planner must be able to plan routes to different goals while taking into account rendezvous based off of resource expenditure.

Finally, communications constraints are frequently applied to existing paths, changing only the velocity of the rover along the path to achieve the constraint. Potential fields that draw rovers together during search are a much more applicable approach, as the planner can generate plans to different waypoints at run time, rather than relying upon some path. Interestingly, the DPC algorithm can be adapted to enforce communication constraints by specifying a continuous distance constraint. In this sense, it would act similar to a potential field, drawing the rovers together if they stray too far apart. While no single solution has all the parts necessary to address the challenges inherent to Symbiotic Exploration, portions of each approach can be combined and adapted to develop a workable solution.

Chapter 3

Methodology

This chapter details the approach this research has taken towards addressing the problems of symbiotic exploration, as well as the methods and metrics used to manifest and evaluate that approach. A new framework for modeling the symbiotic constraints is presented that allows for complex relationships to be simply defined and applied. This framework was implemented in the form of a highly modular multi-robot planner. The experiments are used to evaluate this planner, including parameterization and robot configuration are described in detail. Finally, the metrics to evaluate the planner are presented.

3.1 Approach

The approach towards addressing the problems of symbiotic exploration evolves from many of the methods discussed in Section 2. The DPC algorithm[53] described by Bhattacharya is modified to operate with far more complex relationships and constraints than the original algorithm was capable of through a Tanker-Feeder approach. Within each DPC iteration, A* is used to plan between nodes. The planner that was developed is capable of developing long-duration plans on highly dynamic environments, in part due to a novel storage method that significantly improves the speed at which large amounts of temporal data can be evaluated.

3.1.1 Distributed Planning

Distributed Path Consensus provides the base algorithm for the symbiotic planner. Recall from Section 2.3.2 that DPC takes an iterative approach to the problem of multi-robot rendezvous. After calculating the shortest path to their goals using A*, pairs of robots are given constraints at different timesteps to be within a specific distance of each other. The algorithm then recalculates the path (π_{robot}) for each rover using A*, using the same costs and heuristics as before. Once it expands a node with a timestep corresponding to one of the constraints, the location of its "partner" robot in the constraint at the same point is time is looked up. Much like in [60], a penalty cost is calculated that is equal to the difference between the distance between the two robots and the desired maximum distance between the two robots at that time ($\phi_{i,j}(t)$). If this value is less than zero (i.e. the robots are closer than the maximum distance), the penalty is set to zero. Otherwise, the penalty is weighted by a value ω and applied to the node as an additional cost.

The algorithm then continues its search towards the goal, applying penalties through the same method mentioned above for each node at a timestep where a distance constraint exists. The same is then done for all other robots, and repeated until a solution is found. Between each calculation, the value of ω is increased by a small amount, ϵ. This has the effect of causing the robots to slowly converge on a solution where the robots meet in the middle of their original paths, rather than one

robots path immediately jumping right next to its partner. Algorithm 2 shows the process in its entirety.

Algorithm 2 DPC Algorithm

function PENALTY(π_i, π_j)
  return $\sum_{t=0}^{T} \text{MAX}(0, D(\pi_i(t), \pi_j(t)) - \phi_{i,j}(t))$
end function
for all r $\in$ robots **do**
  $\pi_r^0 \leftarrow argmin_{\pi_r} c(\pi_r)$
end for
$\omega, iter \leftarrow 0$
do
  for all r $\in$ robots **do**
    $\omega \leftarrow \omega + \epsilon$
    $\pi_r^{iter+1} \leftarrow argmin_{\pi_r}\{c(\pi_r) + 2\omega \sum_{i=1}^{\#_{robots}} \text{PENALTY}(\pi_r^{iter}, \pi_i^{iter})\}$
  end for
  $err \leftarrow \sum_{i=1}^{\#_{robots}} \sum_{j=i+1}^{\#_{robots}} \text{PENALTY}(\pi_i^{iter+1}, \pi_j^{iter+1})$
  $iter \leftarrow iter + 1$
while $err > maxError$

A key benefit to using DPC as the core algorithm is that it can address both the resource-based rendezvous and the maximum separation constraints required by symbiotic exploration, albeit with some modifications that are discussed in the following section. The maximum separation distance constraint can be achieved by specifying a distance constraint of some $\phi_{i,j}(t) > 0$ be applied for the entire duration of the path. The effect of this is to ensure that the robots are within that distance of each other at all times. This is extremely convenient, as it means the same underlying algorithm can be used to achieve both symbiotic constraints.

As mentioned before, however, the DPC algorithm as it currently exists is not capable of addressing the symbiotic constraints. The constraints must be defined within the domain of time. In the case of a rover expending energy, the moment in time when it will require recharging is not known. Furthermore the value at which the constraint is applied must be identical across both rovers. There is no way of having the robots rendezvous when one of them has expended a certain amount of energy, regardless of when or where that occurs.

3.1.2 Tanker-Feeder Model

This research addresses the aforementioned challenges of DPC by reformulating the problem and redefining the constraints. The result is a symbiotic planning algorithm that vastly increases the capability of the planner and is able to easily represent complex constraints. Rather than defining a symbiotic constraint as a maximum separation distance and a time at which two robots should be at that distance (as is the case with DPC), a symbiotic constraint is defined as a maximum separation distance and a tuple of sub-constraints, each belonging to a single rover. Each of these sub constraints (ρ) contains a resource determinant (time, distance traveled, energy expended, etc.), a constraint class using the terminology from [52] (Feeder or Tanker), and, if the constraint class is "Feeder", a value for when the constraint should be applied.

While adding in new constraint determinants was necessary for allowing resource-based rendezvous, separating constraints into Feeder and Tanker classes is what enables the complex relationships to form. Specifying a constraint as a Feeder causes the constraint to apply when its set of conditions is met (eg. the robot has traveled a certain distance or has a certain battery level). Tanker constraints are the opposite, applying the penalty when their partner is triggered.

This new approach is realized by substituting the PENALTY function seen in Algorithm 2 with a new one, shown in Algorithm 3, hereby referred to as PENALTY.TF. Given a path for a robot (π_i), the PENALTY.TF function will go through all states (s_i) along that path. A state contains both the physical location, as well as "cost" information (distance traveled, time, energy level, etc.).

At each of these states, a conditional function IS_TRIGGERED is called on that state and and the constraint for that robot (ρ_i). The purpose of the IS_TRIGGERED function is to determine whether or not the robot is currently in a state where its constraint should be applied. For Feeder constraints, this occurs when value for the constraint is equal the value of the constraint determinant within the state. For example, if ρ is a Feeder constraint with a value of 10 meters and with distance as a determinant, IS_TRIGGERED will return true if and only if the distance traveled by the robot at that state is equal to 10 meters.

Algorithm 3 Tanker-Feeder Penalty Function

```
function IS_TRIGGERED(s_i, ρ_i, π_j, ρ_j)
    if ρ_i(class) = FEEDER then
        return ρ_i(value) = s_i(ρ_i(determinant))
    else if ρ_i(class) = TANKER & ρ_j(class) = FEEDER then
        s_j ← π_j(ρ_j(value), ρ_j(determinant))
        return s_i(ρ_i(determinant)) = s_j(ρ_i(determinant))
    else if ρ_i(class) = TANKER & ρ_j(class) = TANKER then
        return TRUE
    end if
end function
function PENALTY.TF(π_i, π_j)
    e_total ← 0
    for all s_i ∈ π_i do
        if IS_TRIGGERED(s_i, ρ_i, π_j, ρ_j) then
            if ρ_j(class) = FEEDER then
                s_j ← π_j(ρ_j(value), ρ_j(determinant))
            else if ρ_j(class) = TANKER then
                s_j ← π_j(s_i, ρ_j(determinant))
            end if
            e = MAX(0, D(s_i, s_j) − φ_{i,j})
            e_total ← e_total + e
        end if
    end for
    return e_total
end function
```

If the constraint for the robot Tanker, then the problem is more complex, as it needs to know about the state of its partner. In this case, the partners constraint also matters. If the partner constraint is also of the Tanker class, than the constraint will always be triggered, for reasons that will be discussed later on. If the partner constraint is a Feeder that the state at which that constraint applied to the partner must be looked up in its path. This is done by traversing the states in the partner's path and finding when the value of partner constraint is equal the value of the partner constraint's determinant within the state. For monotonic costs, such as time and distance traveled, this can be done through a fast binary search. Once the state at which the partner's constraint is applied is found, it is then compared against the main robot state. If the value of the main state using the main constraints determinant is equal to the value of the partner's state also using that determinant, then the constraint is said to be triggered.

While this process seems complicated, the underlying behavior can be shown easily with some

examples. Assume the constraint relationship described in equation 3.1. In this scenario, there are Feeder constraints on both robots to rendezvous after 100 seconds. To figure out whether or not a constraint should be applied for a given state for the main robot it checks whether or not the the time for that state is 100 seconds. If it is, it checks where its partner is when its side of the constraint is met (in this case, also 100 seconds). If the distance between the two robots at those states is 0, a penalty is applied. The original DPC implementation mimics this type of constraint: a Feeder-Feeder constraint where the determinant for both constraints is time, and the values are identical.

$$\begin{aligned}
\rho_{main} &= \{determinant = \text{TIME}, class = \text{FEEDER}, value = 100s\} \\
\rho_{partner} &= \{determinant = \text{TIME}, class = \text{FEEDER}, value = 100s\} \\
\phi_{i,j} &= \{\rho_{main}, \rho_{partner}, distance = 0m\}
\end{aligned} \tag{3.1}$$

Through the Tanker-Feeder model, however, there is no requirement that the values be identical. If the value for $\rho_{partner}$ in Equation 3.1 were instead set to something different, such as 120 seconds the effect would be a "delayed rendezvous", where the two robots travel to the same location, but at different times. In this case, the partner robot will be drawn towards the same location at t=120 seconds where the main robot was at t=100 seconds. Such a constraint could be useful for pick up and drop off scenarios. Furthermore, there is no need that the determinants match. In a scenario such as the one displayed in Equation 3.2, the main robots constraint uses distance as a determinant (with a value of 30 meters), while its partner uses time as its determinant. This constraint would cause the main robot to be at the same location when it traveled 30 meters that its partner was at t=100 seconds, even if the two do not occur at the same point in time.

$$\begin{aligned}
\rho_{main} &= \{determinant = \text{DISTANCE}, class = \text{FEEDER}, value = 30m\} \\
\rho_{partner} &= \{determinant = \text{TIME}, class = \text{FEEDER}, value = 100s\} \\
\phi_{i,j} &= \{\rho_{main}, \rho_{partner}, distance = 0m\}
\end{aligned} \tag{3.2}$$

Another type of relationship occurs when one of the robots is a Tanker and the other is a Feeder, shown in Equation 3.3. Here, the main rover has a Tanker constraint with a time determinant, while its partner has a Feeder constraint with an energy determinant and a value of 100 units. To figure out whether or not ρ_{main} is triggered, the algorithm will search along the partner robot's path for the state where its energy level is 100 units (ie. the partner constraint is triggered). It will then take the time at which that constraint occurs and compare that to the main robot state that is being evaluated. If they match, then the constraint is triggered. The effect of this will be to cause the main and partner robots to rendezvous when the partner robot's energy level is at 100 units.

$$\begin{aligned}
\rho_{main} &= \{determinant = \text{TIME}, class = \text{TANKER}\} \\
\rho_{partner} &= \{determinant = \text{ENERGY}, class = \text{FEEDER}, value = 100\} \\
\phi_{i,j} &= \{\rho_{main}, \rho_{partner}, distance = 0m\}
\end{aligned} \tag{3.3}$$

The final relationship that is possible through the Tanker-Feeder approach is through a Tanker-Tanker constraint. In these relationships, the constraint is always active and is thus always pulling the robots towards each other. Equation 3.4 shows an example Tanker-Tanker relationship, where the determinant for both rovers is time, and the maximum desired is 10 meters. Assume the algorithm is evaluated a state that has a time component of X seconds. When the constraint is evaluated for that state through PENALTY.TF, the the value for the partner's constraint determinant

within the current state is captured (in this case, X seconds). The partner's last computed path is then searched for the state that is closest to t=X seconds. The difference in location between the two states is then used to compute the penalty. If that difference is greater than 10 meters, a penalty will be applied drawing the robots closer together. The effect of this is to ensure that the robots are within 10 meters of each other at all times, addressing the maximum separation distance constraints.

$$\begin{aligned}
\rho_{main} &= \{determinant = \text{TIME}, class = \text{TANKER}\} \\
\rho_{partner} &= \{determinant = \text{TIME}, class = \text{TANKER}\} \\
\phi_{i,j} &= \{\rho_{main}, \rho_{partner}, distance = 10m\}
\end{aligned} \tag{3.4}$$

Distance can also be used as a determinant. Assume the constraint is the same as above, with the exception that the determinants for both are distance rather than time. In such a scenario, the penalties applied would be such that, when one rover has traveled X meters, it will be within 10 meters of where its partner was when it traveled X meters.

It is important to note that robots may have multiple constraints active at the same time. A common configuration in symbiotic planning would be to have a Tanker-Tanker constraint with a time determinant (to ensure the robots stay within communication range) and a Tanker-Feeder constraint on energy (as shown in Equation 3.3) to ensure rendezvous based off of energy usage.

The penalties themselves are computed much the same way as in the original DPC algorithm. If it is determined whether or not a constraint for a given state should be applied, the partner's corresponding state must be found. If the partner constraint class is a Tanker, the partner's path must be search for the corresponding state. This can be done quickly through a binary search, as the paths are, by construction already ordered by distance and time. This search must occur because the Tanker constraint has no fixed state at which it occurs, it is entirely dependent upon its partner's state at the time.

For constraints where the partner is a Feeder, the state at which the partner's Feeder constraint occurs can be cached for even quicker access, as it will only be updated when the partner recomputes its path. For monotonic determinants such as distance and time, only a single value needs to be stored. For non-montonic determinants such as battery life, the process is slightly more complicated. For such determinants, all the states that they occur will need to be stored. This list of states can then be binary-searched for the state closest to the current time.

Once the partner's state is found. the distance between the two robots at those states is computed. If the distance between the two is greater than the desired maximum distance by that constraint, a penalty is applied proportional to that difference. As with the original DPC algorithm, this penalty will be weighted by a value ω that will be increased by ϵ at every DPC iteration. This will cause it slowly converge upon an solution if it exists.

3.1.2.1 Forward Cost Projection

While the above approach will correctly apply the symbiotic constraints, the method by which it will apply them is not ideal. This is because, as in the original DPC algorithm, a penalty is only applied when the constraint conditions for either robot are exactly met (eg. the rover has traveled exactly 300 meters or exactly 500 seconds has elapsed). Thresholding can be applied so the rendezvous will occur if within a certain percentage of the constraint condition, but even that will lead significantly backtracking during the search due to the constraint penalty suddenly increasing the cost of all surrounding nodes.

Figure 3.1 shows how and why this occurs. As the orange robot is developing its plan, the most direct path towards the goal is to drive directly ahead. When it finally gets to $t = 4$, the DPC constraint suddenly causes the node directly in front of it (as well as the one up and towards the

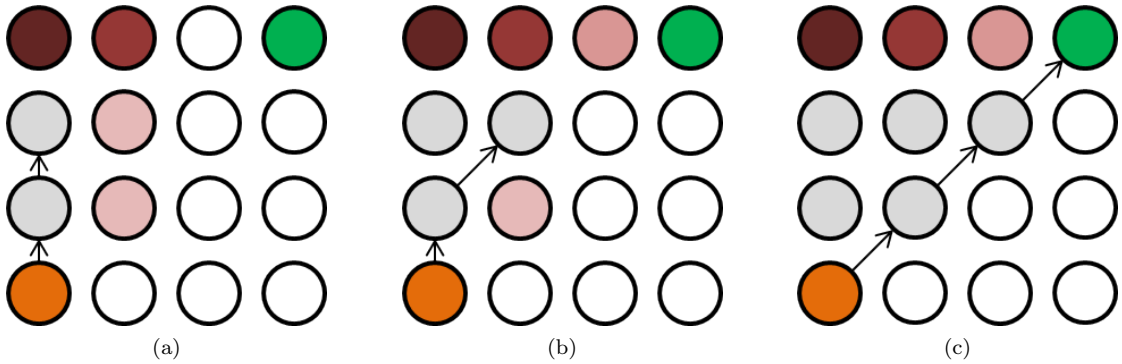


Figure 3.1: State expansion without FCP. The orange circle represents the starting location of the robot being planned for, while green circle shows the location where the partner robot is when the constraint should be applied ($t = 4$). Circles are colored according to their cost, with darker circles costing more. Gray circles have been visited and added to the closed set, while white circles have not been expanded. The x axis represents distance and the y axis represents time.

right) to increase significantly in cost. As all the nodes at $t = 4$ now have a higher cost than those at earlier times, it must backtrack its search, as shown in Figure 3.1b. After repeating this process many times, a valid solution is found (Figure 3.1c), although many more nodes are both expanded and visited than ideally would occur.

This problem is similar to the one faced by uninformed search algorithms, where states are often expanded and visited in a highly inefficient manner because they lack prior knowledge about the search space. Fortunately, the same methods used to address those problems can be applied to this scenario, namely the use of informed heuristics to draw the planner closer to the intended point of rendezvous. These heuristics, referred to as Forward Cost Projection (FCP), act as an estimate of the future symbiotic penalty for the constraints on the rover.

Implementing FCP within the PENALTY.TF function is simple. First, the IS_TRIGGERED function is modified as shown in Algorithm 4 to take into account a threshold. Rather than simply checking whether or not the determinant cost at a state is equal to the constraint's determinant cost, the function instead checks whether or not it is within a a certain percentage (given by $\delta_{threshold}$) of that value. Increasing or decreasing $\delta_{threshold}$ tunes how early the constraint penalties should be applied. Tanker-Tanker constraints, where the constraint is always applied, remain unchanged.

Algorithm 4 Triggering Function with Thresholding

```
function IS_TRIGGERED.THRESHOLD(s_i, ρ_i, π_j, ρ_j)
    if ρ_i(class) = FEEDER then
        V_threshold = ρ_i(value)δ_threshold
        return ABS(ρ_i(value) − s_i(ρ_i(determinant))) ≤ V_threshold
    else if ρ_i(class) = TANKER & ρ_j(class) = FEEDER then
        s_j ← π_j(ρ_j(value), ρ_j(determinant))
        V_threshold = ρ_j(value)δ_threshold
        return ABS(_i(ρ_i(determinant)) − s_j(ρ_i(determinant))) ≤ V_threshold
    else if ρ_i(class) = TANKER & ρ_j(class) = TANKER then
        return TRUE
    end if
end function
```

The states for both robots are then found, as per the original PNEALTY.TF algorithm. The only other change that needs to be made is how the "error" penalty is calculated. Rather than simply subtract the distance between the two states by the desired distance at the constraint, we need to add an additional offset that represents how on track the rover is to meeting the constraint. The process for computing this offset is shown in Algorithm 5.

Algorithm 5 FCP Offset

```
function FCP_OFFSET(s_i, s_j, ρ_i)
    if ρ_i(class) = FEEDER then
        Δ_c ← ABS(s_i(ρ_i(determinant)) − ρ_i(value))
    else if ρ_i(class) = TANKER then
        Δ_c ← ABS(s_j(ρ_i(determinant)) − s_i(ρ_i(determinant)))
    end if
    return h(Δ_c)
end function
```

First, the cost differential (Δ_c) between the current state and when the constraint should occur is computed. In the case of a Feeder constraint, this is simply the magnitude of the difference between the constraint value and the determinant cost of the current state. For Tanker constraints, it is the magnitude of the difference between determinant cost of the partner robot's state, and the determinant cost of the current state. Once this differential is computed, a heuristic (h) is applied to it to determine the offset. This heuristic estimates the distance to get from the current state to a position where the constraint will be met. If the constraint determinant is distance, Δ_c can be returned directly. For time as a determinant, an admissible heuristic would be to multiple Δ_c by the maximum velocity of the rover. The resulting penalty can then be found through Equation 3.5.

$$e = \text{MAX}(0, D(s_i, s_j) - (\phi_{i,j} + \textsc{fcp\_offset}(s_i, s_j, \rho_i))) \tag{3.5}$$

The result is an effect similar to the one shown in Figure 3.2. Note that, because the constraints are applied at the beginning of the search, the planner is drawn towards the location where the constraint should occur. This results in it expanding and visiting far fewer nodes than without Forward Cost Projection. While there is an added cost in computing the constraint costs at every node expansion, the resulting decrease in overall node expansions and visitations should counteract this increase, particularly on much larger maps where plans without FCP may require significant backtracking.

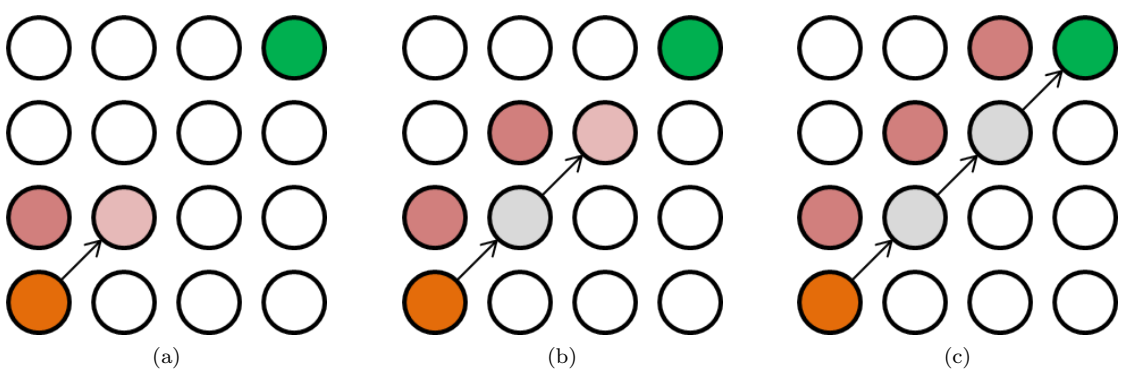


Figure 3.2: State expansion with Forward Cost Projection.

3.1.3 Hazard Constraints

The planetary landscape is rife with hazards. Precipitous slopes can present challenges to rovers incapable of traversing such terrain. Communication shadows can prevent data from being sent to and from rovers. And darkness can prevent the rover from recharging its energy stores and poses thermal challenges on the rover. The planner must take into account these hazards when developing its plan, or the resulting plan may lead to the rovers demise.

Some of these hazards, such as slope, are static and can easily be preprocessed to ensure fast traversal through the graph. Others, such as communications and lighting vary with respect from time, making them far more challenging to process. Commonly, a lookup table for each of these hazards (mapping the time to the "value" of that hazard will be generated) for each node. At run time, if the state of the node at a given time is needed, that time is simply indexed into the lookup tables, and the hazards returned.

While the lookup table method is fast, it suffers major performance degradation when dealing with long periods of time and large maps. This is because the memory required to store the map grows linearly with respect to the number of time increments. For example, assume the map is a 1000x1000 grid with both the communications and lighting maps being stored. Every time step that is added to the lookup table introduces 2 million new elements that must be indexed. When dealing with the multi-month missions the symbiotic scenario seeks to operate on, the memory required quickly becomes difficult to manage, particularly for a small rover. Worse even with a large amount of memory, cache effects will slow down the planning processing significantly, as the planner can no longer easily access sequential blocks of memory.

Depending upon the application, the timesteps can be binned into coarser steps if the map does not change significantly between steps. In the symbiotic scenario, however, it is likely that some portions of the map will change rapidly, particularly around the edges of craters, so losing temporal resolution is a non-option.

To address this challenge, a novel approach to map storage is used. Rather than densely storing each hazard value for all nodes at every time step, only the first timestep is densely stored. At subsequent timesteps, the value for a hazard within a node will only be registered if it changes by a certain amount relative to the last stored value. This operates in a fashion to video compression algorithms, where an infrequent keyframe is densely stored, and subsequently only differences between the images are saved.

Thus, rather than saving each hazard within a node as an array of t values, where t is the number of timesteps, it will instead be stored as an array of tuples, where each tuple consists of the timestamp of the "transition" as well as the value at the transition. Because this array will inherently be sorted by time, if the value for a hazard at a specific timestamp is desired it can be found quickly using binary search. While the performance will not be as fast as the $O(1)$ lookup time for the lookup table, the binary search will have a worst-case performances of $O(\log n)$ (where n is the number of transitions for that node/hazard), enabling it to scale well with the number of time steps. And while some regions will have many transitions, others will transition infrequently. Some, particularly the perma-dark regions that symbiotic exploration is interest in exploring, will not transition at all, meaning access is just as fast at with a lookup table, without wasting storage space.

Algorithm 6 shows the binning process for a single hazard and a single node. Every node has an array H for each hazard containing the hazard transitions. For each time step, the value for that hazard at that node is grabbed. If it is less than some minimum bin size (B_{min}) it is set to a minimum value (V_{min}). If it is greater than some maximum bin size (B_{max}) it is set to a maximum value (V_{max}). Otherwise, the bin it belongs ($b_{\#}$) to is computed by normalizing the valid ranges to be 0 min, dividing it by the bin size (B_{size}), and then flooring it. Once the bin number is found, it is then converted back to the appropriate value for the bin. If this value differs from that of the last element in H, the time value pair is appended to the end of H. This process is then repeated

for each node and each hazard. It is important to note that each hazard may have different bin parameters (B_{size}, B_{min}, V_{max}, etc.).

Algorithm 6 Hazard Binning Function

```
H ← ∅
for all (t, v) ∈ (times, values) do
    if v < B_min then
        v ← V_min)
    else if v > B_max then
        v ← V_max)
    else
        b_# ← FLOOR((v − B_min) / B_size)
        v ← b_# B_size + B_min
    end if
    if H[−1].v = v then
        PUSH(H, t, v)
    end if
end for
```

Having the additional minimum and maximum bin and value parameters versus just the bin size is a simple optimization that allows for much greater flexibility and storage space reduction. For some hazards, such as communications, any value lower than a certain amount is unusable for communications and anything greater than a different value is overkill. Within those two values, however, high resolution is needed to determine whether or not the signal strength would be strong enough. Simply specifying a small bin size could yield too many bins and cause performance issues. By specifying these additional parameters, the effect is to have high resolution where it is needed and low resolution where it isn't. Indeed, if the hazard being binned has a hard threshold and no soft threshold, it should effectively be treated as binary: either you can travel through it or you cannot. Setting the bin parameters accordingly will cause only transitions between when the node is passable or unpassable to be stored in H, making it much quicker to traverse when looking up the value for a given time.

Besides the significant performance benefits entailed by using this sparse method of storing hazard data, it also exposes information about the hazards that would otherwise be difficult or time consuming to calculate. This information can then be leveraged during the planning stages for faster/better results. For instance, it may be valuable to know just how long a rover can stay in a location before it must move due to the hazards in that region causing it to be invalid. In a dense storage solution, every time step from that point in time on must be checked. Compare this to the binning solution, where a significantly compressed lookup needs to be checked.

3.2 Experimentation

This section describes the experimental approach for validating the planner. It includes types of maps that plans will be generated for, the various robot parameters that will be applied to change path output, as well as the symbiotic constraints that will be applied to the robots during the plan.

3.2.1 Maps

Every planner must have a map that it operates on. The planner will be evaluated against two distinct types of maps: synthetic maps and real-world maps. Synthetic maps are manually generated

to fit a specific type of experiment, and offer very little real-world analog, while real world maps are taken directly from real data of the Moon.

3.2.1.1 Synthetic Maps

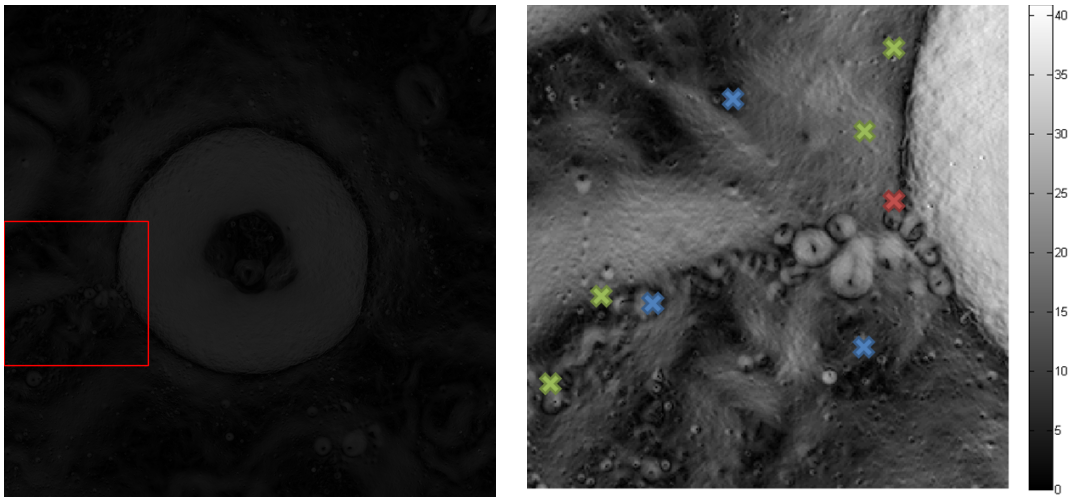
Although synthetic maps have no real-world analog, they provide are useful for validation of the planner. Maps generated using real data are often complex and unpredictable, making it difficult to determine whether or not the planner is behaving as expected or if the generated path is ideal. Manually creating maps allows for each environment to be tailored to the particular experiment being run.

Validation that algorithm behaves as expected is conducted on a blank world map. A blank world map is, as the name implies, a completely empty world. In the case of the symbiotic planning, this means the slope of all cells in the map are set to 0°and lighting maps are such that the entire region is fully lit. Communications is likewise assumed to be at full strength at all locations. Although such an environment is wholly unrealistic, it is particularly useful for determining whether or not path output is reasonable for a set of symbiotic constraints. Given that the conditions are both known and ideal, the planner should have no difficulty finding a perfect solution if one exists.

3.2.1.2 Lunar Maps

While synthetic maps offer easy validation of the planner, they will not offer viable plans that can be used. For real utility, maps of prospective Lunar sites must be used. The maps discussed below were generated from digital elevation models (DEM) taken from the Lunar Reconnaissance Orbiters (LRO) Lunar Orbiter Laser Altimeter (LOLA). These DEM's have resolution ranging from 2-20 meters per pixel.

The DEMs were then used in conjunction with the Sun's ephemeris to generate lighting maps over time through a ray-tracing algorithm. Similarly, maps of regions where the rover can com-



(a) Shackleton Crater slope map. The boxed region indicates the area planning will operate within.

(b) Zoomed in region on Shackleton Crater, color-adjusted for slope, with areas of interest.

Figure 3.3: Slope maps of Shackleton Crater. In subfigure (b), the red cross indicates the starting point. Blue crosses indicate permanently dark regions of interest, while green crosses indicate areas of interest that experience mixed lighting.

municate to Earth at a given point in time can be generated from the DEMs. The lighting and communications maps can be generated for a periodic number of timesteps to create a 4-D model of the lunar environment to plan upon. Each map will be evaluated using multiple starting locations and goal locations corresponding to prospective landing sites and areas of interest. The key region of interest for experimentation on the Lunar landscape is be Shackleton Crater.

Located almost exactly on the Lunar south pole, Shackleton Crater has regions that are enveloped in permanent darkness, as well as areas that experience almost continuous sunlight [62]. This makes it a valuable location for permanent exploration habitats, as the high level of sunlight lead to relatively high temperature levels and solar power potential [63]. Additionally, recent missions have indicated above average concentrations of hydrogen close to Shackleton, making it a prime target for exploration missions [64].

There are a number of areas of interest that the planner will plan for, marked as crosses in Figure 3.3(b). Some of these areas are in permanent darkness, and will only be traversable by rovers with the capability of operating outside of sunlight. Others have limited lighting, and will be split between the two rovers according to their capabilities.

3.2.2 Robot Parameters

If the maps detailed above describe the environment that the planner will be operating in, then the parameters detail how the planner will interact with that environment. Given the high variability in robot designs and capabilities, great care was taken to ensure a similar level of modularity in the planner. This section will discuss the different parameters that can be modified within the planner, and how they will be used during the testing and evaluation of the planner.

3.2.2.1 Robot Velocity

Each rover is given a list a velocities that it may travel at between two neighboring nodes. Robot velocity plays an important role in path generation, not only because it determines when a rover will reach its destination, but it also affects the energy model and symbiotic constraints of the rover, which will be discussed in subsequent sections.

To limit the search space of the problem, we will discretize the possible velocities. The velocities that will be used are based upon the average and max velocities of the MER, MSL, Yutu, and India's Chandrayaan-2 rover, and will range from 1-20 centimeters per second. The entire range of velocities will not be available to every rover for each experiment. Instead, the experiments will vary the velocity range for each rover to assess what configurations offer the best performance.

In much of the experimentation, the secondary rover will have a higher velocity than the primary rover. This is due to the high likelihood that the secondary rover will need to cover a much greater distance than the primary rover in the same amount of time if it is to explore remote areas while returning to the primary rover on time. Experiments will, however, be run with both rovers having equivalent velocities and with the velocity ranges reversed from the above scenario in order to test whether or not this hypothesis is true.

3.2.2.2 Energy Model

The energy model is not a parameter, but an equation with a number of parameters that can be modified. Equation 3.6 shows a parameterized version of this equation. The energy model is used for determining whether a rover has enough battery power to cross into a shadowed region, or if it must instead sit and recharge. If a rendezvous constraint based upon a rover's energy level is specified, the energy model will also be used. Experiments will be conduction assuming different energy model parameters for each rover, as detailed below.

$$E_{i+1} = \max(E_{max}, E_i - (\rho_{hotel} + \rho_{motor} - \rho_{solar} - \rho_{other})\Delta_t) \tag{3.6}$$

Where:

E_{max}: The maximum allowed energy level (ie. the battery capacity)

ρ_{hotel}: The base power draw of the rover

ρ_{motor}: The cost of driving the motor

ρ_{solar}: The power generated from solar panels

ρ_{other}: Charging/power draw by other sources (ie. the primary rover charging the secondary rover)

Battery Capacity The battery capacity, coupled with the rate of power expenditure, controls how long a rover can last without an external power source. If the rover lacks solar panels strong enough to sustain the rover indefinitely, or if the rover is traveling through shadowed regions, high battery capacity is critical to achieving a lasting mission duration and far rover range. Battery capacities between 4-6 kWh will be evaluated to see how they affect the constraints, paths, and what goals can be traveled to.

Hotel Cost The hotel cost of a rover is the power required to keep the rovers electronics powered on, assuming the rover is stationary. This includes cameras, CPU, communications, and other electronics. It is essentially the minimum amount of power (excluding recharging) that the rover may draw at any point in time. Though the hotel cost of the rover should be relatively stable, it can be increased or decreased by adding or removing electronics and scientific instruments. For instance, an antenna powerful enough for direct communications to Earth requires roughly 90 Watts of power[65] . Such an antenna will generally not be necessary for a secondary rover in the symbiotic scenario if it can relay communications through the primary rover or lander. During experimentation the following electronic/instrument configurations with a power consumption between 75 and 150 Watts will be explored.

Motor Cost The amount of power allocated for motors plays a significantly role in the capabilities of the rover. Assigning values a static cost to the motor is difficult, because it is almost entirely dependent upon the overall configuration of the rover. By assuming a worst case scenario (maximum velocity up maximum slope) the maximum required power of the motors can be determined. Among the motors with a power draw sufficient to meet the maximum requirement, the most efficient motor operating at the average expected velocity/slope can be selected. Because motor power draw is so dependent upon other parameters, this parameter will not be manually tuned, but instead be automatically assigned based off of the rest of the configuration.

Solar Recharge Solar recharge is similarly dependent upon a number of parameters, as seen in Equation 3.7. For the sake of simplicity, experiments will assume that ψ_{sun} is always 90°(ie. the solar panel is always perpendicular to the suns rays). Expected efficiency of solar panels on planetary missions is expected to be roughly 30%. Solar flux dependent upon both the environment (atmosphere, distance from the sun, etc.) and lighting conditions (whether or not the region is shadowed at a given point in time), and will therefore be determined based off of the lighting maps. Unfortunately, the intensity value from the lighting maps is not an accurate gauge of the actual level of solar flux at a given location. Instead, a binary flux is assigned such that, if the intensity is a cell is above a certain value, that cell is assumed to be fully lit, otherwise it is darkened.

$$\rho_{solar} = \psi_{flux}\sigma_{panel}A_{panel}\cos\phi_{sun} \tag{3.7}$$

Where:

ρ_{solar}: The power generated by the solar panel

ϕ_{flux}: The solar flux at the location

σ_{panel}: The conversion efficiency of the solar panel

A_{panel}: The area of the solar panel

ψ_{sun}: The angle between the sun's rays and the solar panel normal

External Recharge The final component of the energy model comes from external charging factors. This variable includes all other charging and discharging influences not mentioned elsewhere. This may be recharging from an RTG (as could be the case for the primary rover), being recharged from another rover entirely (as might happen if the secondary rover recharges from the primary rover), order the swapping out of a battery. In the latter instance, this value could act as a drain on power from the primary rover if high enough, and it may be required to tap into its own battery stores.

The key dependencies of this parameter that will be explored are RTG power supply and multi-rover recharge rate. If a rover is powered from an RTG, it will have a greater capability to explore darkened areas without fear of draining its batteries. This benefit will have to be balanced against the increased cost of an RTG. Experiments will be conducted exploring how much value is lost if an RTG is not used in lieu of solar panels. The second parameter that will be tuned for is multi-rover recharge rate. Higher rates mean faster recharging, but may also prove to be too great a power drain for the rover supplying power.

3.2.2.3 Hazard Thresholds

A key feature of symbiotic exploration is its ability to mitigate risk. Accounting for the various hazards in the planetary landscape is necessary to assess the risk of the environment. Rover capability for hazards will be assigned through the use of "hard" and "soft" thresholds for each hazard explored. A hard threshold is one where, if the risk at a given location passes beyond that threshold, that location is considered off limits. An example of this would be a precipitous slope that would prove fatal to any rover trying to traverse it. In these situations no path should ever have the robot travel through such a region.

Soft hazards are different in that the hazardous region is not considered off limits, but a penalty will be assigned during path generation that will push the rover away from such regions unless extremely beneficial. For example, it may be more beneficial for a rover to go over a slightly steep slope if the alternative is a long trek around a hill that would detract from the mission. Such penalties can be assigned as a constant value (eg. always apply a specific penalty if the lighting is below a certain level), or be a function of the hazard itself, for instance, a higher penalty for a steeper slope.

Although the risks inherent to planetary exploration are near limitless, this research will focus primarily upon three main hazards: slope, shadows, and lack of communication. All three were generated through the methods discussed in Section 3.2.1.2.

Slope Slope as a hazard differs from both shadows and communications in that it is a static hazard. This allows for some optimization to be made when considering it, as we can remove any node with a slope greater than the hard threshold from being explored without having to account for time. The slope maps generated from the DEMs contain the angle of the terrain in degrees,

with the maximum slope saturated at 30 °. This saturation was done to address noise artifacts in the DEMs that resulted in extremely large slope values at some locations.

During experimentation, hard slope thresholds of 15°- 30°will be applied for the rovers (though both rovers will not necessarily have the same slope threshold for the same experiment). Additionally, there will be three different soft threshold configurations that will be tested: no soft threshold (all slopes less than the hard threshold will be treated as free space), a linear penalty with respect to slope, and penalty that increases with respect to the sin of the slope angle.

Shadows Shadows are a dynamic hazard in Lunar environments. Darker regions are often colder (which presents challenges for electronic if the temperature drops too low for extended periods of time) and require non-vision oriented sensing to safely traverse. Additionally, they amount of shadow in a region influences the energy level of rovers if it is powered by solar panels.

Like testing for slope, experiments will be conducted using a wide range of hard and soft thresholds for both rovers. In keeping with the symbiotic scenarios posed in Section 1, most of these experiments will assume that the primary rover either have a very low hard threshold on the amount of shadow in an environment or a high penalty associated with the soft threshold for shadows. This will cause it to tend towards traveling in the safer, well-lit regions of the Moon unless absolutely necessary. Conversely, the secondary rover will have a much higher hard threshold (if one will be assigned at all), and a lower penalty in the soft thresholds. Indeed, in some of the experiments such penalties will be removed entirely, as the goal may be to investigate permanently shadowed regions within craters.

Communications The final hazard that will be experimented upon is communications or, more accurately, the lack thereof. Communications shadowing presents a challenge to operators attempting to remotely control the rovers. Without a communications link to Earth, they cannot command the rover or receive updates about its status. A rover failure occurring in an area devoid of communications can be a mission ending event.

Communications hazards will be treated differently from both shadow and slope hazards. First because a region will either have a communications link or not, the thresholds applied will be slightly different. In the experiments the primary rover will only be able to travel to and from regions if there is no communications shadow at the time of travel. If a communications shadow passes through the primary rover's location, it will be forced to remain there until communications returns. For the secondary rover, no such constraint will be imposed. The symbiotic scenario already assumes that the secondary rover will not have the capability to communicate directly with Earth, relying instead upon the primary rover to act as a communications relay. Therefore, no penalty will be imposed upon the secondary rover due to communications hazards.

3.2.2.4 Optimizing Functions

The highly iterative approach this research uses for the planning problem, coupled with the large maps and long mission duration necessitates a high level of optimization to efficiently choose the best plan. This section will explore different optimizing functions that will be tested, how they will be modified, and what metrics (discussed further in Section 3.3) they will seek to optimize.

State Dominance A strong state dominance function is essential in pruning an otherwise intractable search space to one more manageable. How effective the state dominance function is at reducing the search space depends upon what metrics are used. The state dominance function proposed by Cunningham through the following procedure[42]:

1. All else being equal, a state with lower time cost will be preferred

2. If two states have the same time cost, the one with the higher energy (ie. lower energy cost) will be dominant

This state dominance function will be modified so as to include distance traveled as the dominant component instead of time. When dealing with the symbiotic constraints, an additional check of the symbiotic penalty cost for a node will have to be added, otherwise states that are required to meet the symbiotic constraints that might get unintentionally discarded. The check would favor states with lower constraint costs (such as might occur if waiting for another rover to arrive). The key metric that the state dominance function will attempt to reduce are the runtime performance metrics of processing time (both wall and CPU), memory usages (by having less states in the search queue at a given time), and the total number of states visited and expanded.

Goal Heuristics There are a number of heuristics that can be used to reduce the amount of nodes the planner must search before finding a goal. Goal heuristics provide a rough estimate of how far away a node is from a goal. By using a heuristic to provide that estimate and adding that as an additional cost for the node during node expansion, the planner will prioritize visiting nodes closer to the goal. A key requirement of heuristics is that they be admissible. That is, the estimate they produce must always either perfectly estimate or underestimate the eventual cost to the goal. If they do not, then there is a chance that the produced plan will not be optimal. However, having a heuristic that underestimates the cost too much may end up providing little benefit, as it is not strong enough to push the search towards the goal.

As the admissibility of a heuristic is dependent upon the component (time, distance, energy, etc.) being optimized, the heuristics that can be used must also vary based off of that component. If the cost component being optimized is distance, than the heursitic must provide an estimate of distance. Given that the maps being operated on are 8-grids, there are three admissible heuristic functions that will be used to estimate distance: Euclidean, Chebyshev, and Octile distances.

A Euclidean distance heuristic treats estimated cost to travel to the goal from a node as the exact distance between that node and the goal. This is calculated through the Euclidean distance function shown in Equation 3.8. A Euclidean distance heuristic tends to work best when the robot is unconstrained by a grid and can move at any angle. As the planner is operating in an 8-connected grid, whether or not the Euclidean distance heuristic underestimates the cost too much will need to be explored.

The Chebyshev distance (shown in Equation 3.9) is defined as the largest magnitude of the differences between any two points (eg. a node and the goal) along any coordinate dimension. In this case of an 8-grid search, it assumes that all movements (diagonal and othogonal) have the same cost. For this reason, it tends to significantly underestimate the distance to the goal when compared to the Euclidean heuristic. While the number of nodes expanded is likely going to be higher than that of the Euclidean heuristic, it does benefit from being much less costly to calculate, requiring only a comparison versus the relatively costly squaring/square rooting used by the Euclidean metric.

The final distance heuristic that will be used is Octile distance. Octile distance (shown in Equation 3.10) is computed by first computing the total distance between the node and the goal that would need to be traveled assuming no diagonal movement could occur (ie. the Manhattan distance). That value is then subtracted by the distance that can be traveled diagonally between the two locations. The Octile distance heuristic is well suited for 8-grid search spaces, and it is expected to perform well.

$$d_x = |n_x - g_x|$$
$$d_y = |n_y - g_y|$$
$$h_{euclidean} = \sqrt{d_x^2 + d_y^2} \tag{3.8}$$
$$h_{chebyshev} = \max(d_x, d_y) \tag{3.9}$$
$$h_{octile} = (d_x + d_y) + (\sqrt{2} - 2)\min(d_x, d_y) \tag{3.10}$$

In addition to distance as the optimizing component, total time of the path will also be evaluated. In this case, the heuristic used will be the estimated distance to the goal, divided by the maximum velocity of the rover. As the distance to the goal is required by this heuristic, the time heuristic will be evaluated using each of the above distance heuristic methods.

Constraint Heuristics As mentioned in Section 3.1.2.1, forward cost projection acts as a heuristic estimate of the future symbiotic constraint cost and guides the rovers towards the rendezvous locations before the actual constraint value is reached. Though this will result in fewer nodes expanded, it is a costly procedure to compute. One parameter for this heuristic that can be tuned is how early the forward projection should be computed. In some cases, it may not be worthwhile to compute the cost projection if the constraint will not be applied for a significant period of time. Experiments will be conducted to find just how early the forward cost projection for a constraint should be applied.

3.2.3 Symbiotic Constraints

Finally, the symbiotic constraints that will be applied to the rovers will be varied and evaluated. The purpose of this is twofold. First, it is to validate that the algorithm does correctly enforce the symbiotic constraints for the duration of the path. Validation will be done primarily on synthetic maps, as it should be much easier to identify whether or not a constraint is truly being met.

The other purpose of testing different symbiotic constraints is to check what parameters are necessary to allow exploration of the most dangerous and scientifically valuable planetary regions. Tuning different constraints will reveal what communications range is necessary for a path to be viable, or what battery capacity is required to enable a rendezvous after exploring permanently dark regions. For this purpose, testing will be computed and reported exclusively on real-world data after the process has been sufficiently verified on synthetic data. All three relationships (Feeder-Feeder, Tanker-Feeder, and Tanker-Tanker) will be tested with varying determinants, constraint values, and distances.

Feeder-Feeder The Feeder-Feeder constraints that will be be primarily based upon distance and time, (ie. enforcing a rendezvous after either a certain amount of time has elapsed or after both rovers have traveled a certain distance). As symbiotic exploration is not as dependent upon Feeder-Feeder constraints as the other two, experiments on Feeder-Feeder constraints will instead attempt to prove that the proposed algorithms work. Additionally, as the Feeder-Feeder constraint is the simplest configuration to verify, validation of different parameters configurations will be used in conjunction with Feeder-Feeder constraints on distance and time prior to their usage in more complicated scenarios.

Tanker-Feeder The Tanker-Feeder constraints that will be evaluated will have the primary rover as a Tanker with a determinant of time, and the secondary rover as a Feeder with an energy-based determinant. The constraint value of rendezvous that the Feeder will attempt to satisfy will be set

between 10 and 30 percent of the total battery life. Experimentation will vary this value and see how much it affects paths. A delicate balance must be maintained, however. Too high a value will limit the secondary rover's exploration capability, as it will need to rendezvous more frequently. Too low a value may prove too risky. Experimentation will also validate the effect that modifying the energy parameters of the secondary rover has on the constraints and paths, specifically increasing and decreasing the maximum battery capacity and solar recharge capabilities.

Tanker-Tanker The final constraint that will be applied is a Tanker-Tanker constraint where both rovers have time as a determinant. This will cause the distance constraint to be applied at all time steps. Maximum distance values between 500 to 2000 meters will be investigated. Ideally, the rovers will remain in communication distance for the duration of the path, although it may not always be possible to do so. If a valid path within the constraints is not possible, paths will be evaluated based upon the maximum amount of time spent outside the communication range.

3.3 Metrics

This section will discuss the metrics by which the above configurations and experiments will be evaluated. They are broken up into four categories: runtime performance, path cost, constraint satisfaction, and robustness.

3.3.1 Runtime Performance

The runtime performance metrics these experiments are aimed at evaluating will test which configurations offer the fastest or lowest cost generation of paths. This is particularly important if the planner is to be run on the low-power computers and FPGAs that most rovers are equipped with, or if the planner is to be adapted to run in real time with updates from the rover. Although all experiments will record the below metrics, they will be most valuable for determining which state dominance and heuristic functions perform best. The metrics that fall into this category are wall/CPU time, states visited/expanded, memory usage, and planner iterations.

Wall/CPU Time This is the most direct measure of how long it takes to generate a path, as it represents the true amount of time spent during generation. Wall time is the real-world time spent generating the plan. CPU time is the amount of time spent in user code by the CPU, not including kernel-level calls or system I/O. The distinction is made because it is possible that much of the code may be parallelized in the future, which could lead to lower wall times and higher CPU times. It is important to note that, while this metric will be useful for comparing the relative performance of different experiments, it should not be used as a definitive representation of how fast a planner will be, as it is affected by a number of other uncontrollable parameters.

States visited/expanded Another measure of planner speed is to measure the total number of states expanded and visited during path search. Ideally, a lower number of states either visited or expanded means lower overall runtime. This is particularly useful for measuring the performance of the state dominance and heuristic functions, as their direct goals are both to reduce the overall number of states searched. Like the Wall/CPU time metrics mentioned above, care must be taken when evaluating performance through this metric. If an optimization reduces the number of states visited, but takes significantly longer to compute, it may not be beneficial to use it. Thus, it is important to look both at the virtual performance enhancement as well as real-world timings when determining what optimization functions improve the runtime performance of the planner.

Memory Usage Although the drawbacks of high memory usage may not be as apparent as slow code, it is important to take it into account, particularly if the code is to eventually run on low-power rovers. Although memory usage is generally a useful tool for determining whether or not bugs exist in the software, it can also be used to evaluate how effective algorithms are at pruning states from the search space. It is expected that memory usage performance will proportional to how many states are expanded.

Planner Iterations The final performance metric that will be measured is the number of iterations required for convergence. In this case, the iterations required are the number of DPC iterations above the initial plan with no penalty. This will likely be affected both by the heuristics chosen (better heuristics may produce more accurate plans faster), as well as the penalty epsilon at each iteration. When adjusting the penalty epsilon, the path cost (discussed in Section 3.3.2) needs to be scrutinized heavily, as higher epsilons likely result in a faster rate of convergence, but may lead to poor paths if too high.

3.3.2 Path Cost

A fast planner is useless if the paths it generates are poor. Thus the optimality of the paths themselves will also be measured. Determining optimality in multi-dimensional problems is challenging. It is often the case that a plan will improve upon another plan in one metric, will performing worse in a separate metric. In such cases, it is difficult to say which plan is truly better. Furthermore, determine true optimality of solutions is sometimes intractable when the search space becomes large, as in the case of the traveling salesman problem. To benchmark the performance of different configurations the path cost metrics will be split into separate metrics, and each configuration will be compared against each other.

Path Distance/Duration The most common performance metric for path planning is which path is the shortest, either in distance or time. In most scenarios, the desire is that the robot do the least amount of extraneous work required to get to its goal location. All else being equal, if one path is shorter than another, it is generally considered the better path. For exploration tasks, this is not always the case, as sometimes there is value to exploring more area if the rover still gets to its end goal within a given time window.

The path distance and time will be measured with respect to the different hazard capabilities, energy models, and velocities used by the rover. It is expected that the greater the hazard capability of the rover, the shorter the path. This is because a more capable rover can cross over terrain that is otherwise intraversable by a less capable rover. Similarly, greater energy capability and velocity are expected to result in shorter paths, although how much they will do so remains to be seen.

Path Value Certain paths may be of higher value than another with otherwise shorter routes. Paths with high value may take a route that passes by interesting terrain on their way to their waypoints, rather than a shorter, yet less interesting route. Similarly, there may be value to waiting at a waypoint for longer than the minimum allowed duration once the rover arrives there. In such a case, the ideal path would be to stay at a waypoint, either drilling for or collecting samples or data or making more observations, until the marginal value gain is overridden by the necessity to travel to a new location.

3.3.3 Constraint Satisfaction

As the goal of this research is to develop a planner capable of addressing the various symbiotic constraints, it is essential that whether or not those constraints are met is monitored. For all parameter configurations, whether or not the symbiotic constraints were met will be recorded. If

they are not met, the error will be evaluated to see whether or not it is acceptable. In some cases, the planner may not be able to find a complete solution due to the discretization of the graph, but will be able to find a solution close enough that the error can be ignored. Additionally, how many iterations of the planner are required to converge upon a solution will also be compared as, all else being equal, finding a solution with fewer iterations is preferable.

Chapter 4

Results

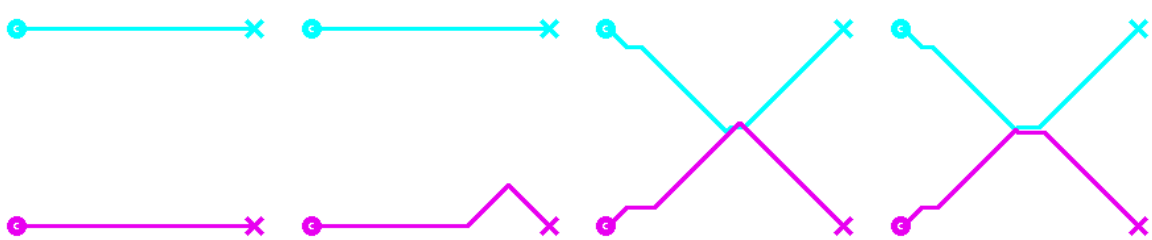
4.1 Symbiotic Constraints

Given the complex nature of the symbiotic constraints, they were initially verified using only the static world maps. This section will discuss the effectiveness of the DPC.TF at enforcing the Feeder-Feeder, Tanker-Feeder, and Tanker-Tanker constraints individually applied on the synthetic maps before evaluating their performance in realistic scenarios.

4.1.1 Feeder-Feeder

Feeder-Feeder constraints apply when both robots are in a state where their respective constraints trigger. Though not necessary in the symbiotic scenario, it is still extremely useful to get rovers to rendezvous at a certain time or after a certain distance has been traveled. In the below examples, both robots have the same capabilities and symbiotic constraints.

The first Feeder-Feeder constraint used a determinant cost of 12000 seconds as the rendezvous constraint for both rovers. Figure 4.1 shows the resulting paths by iteration. Interestingly, it took seven DPC iterations with a penalty ϵ of 1.0 before the penalty weight was enough to draw the rovers away from their paths. This indicates that either a higher starting penalty or ϵ may be beneficial. After 15 iterations, a solution was found that resulted in no error (ie. the robots rendezvoused at $t = 12000$). At iteration 12, there was a rendezvous at roughly $t = 13800$ seconds, but no rendezvous occurred at the specified constraint value, so the planning continued. This can



(a) Initial Paths (b) Iteration 7 (c) Iteration 12 (d) Iteration 15

Figure 4.1: Paths of two rovers with with a Feeder-Feeder time constraint at $t = 12000$ seconds. The circles indicate the starting points while the crosses indicate the goals for each rover. Each subfigure represents the paths after the given number of DPC.TF iterations.

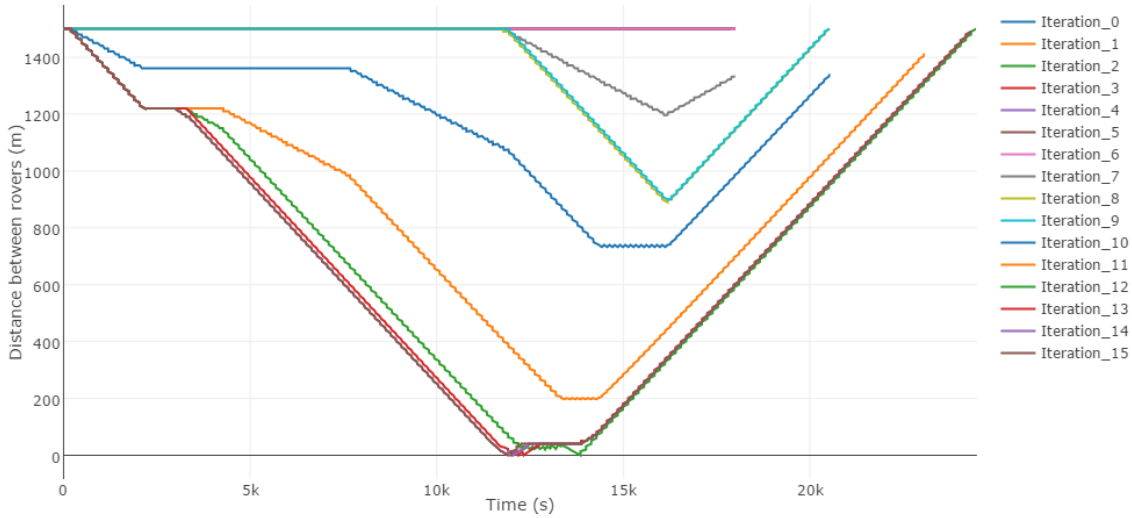
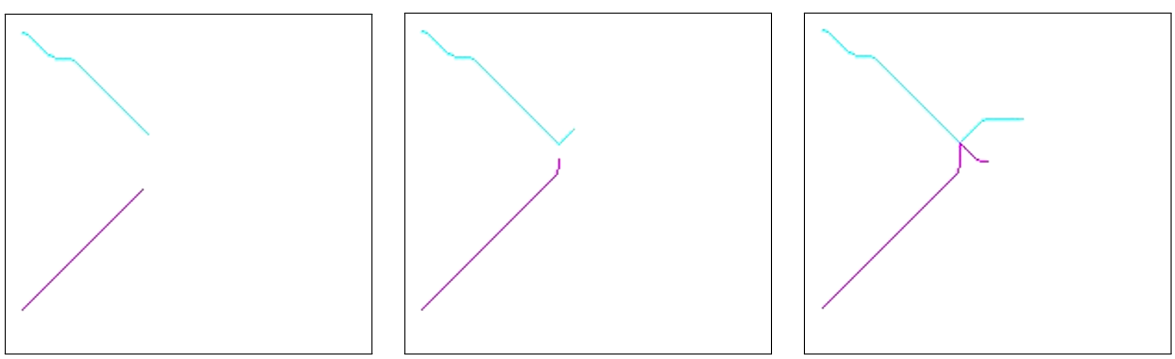


Figure 4.2: Distance between rovers vs Time: Feeder-Feeder ($t = 12000$s)

be seen in Figure 4.2 which shows the distance between the two rovers vs time for each iteration of DPC.

Feeder-Feeder constraints with identical times as the rendezvous value do not truly show of the capability of DPC.TF. Indeed, such constraints are possible with the original DPC algorithm. DPC.TF extends this capability by enabling constraints where the two rovers have different determinants or values to use to rendezvous. Figure 4.3 shows an example of one such configuration, where the magenta rover wants to be at $t = 12000s$ where the cyan rover was at $t = 12000s$. The rovers "rendezvous" at the same spatial location, just at different times.

Figure 4.4 shows the distance between each rover and the rendezvous point over time. Note that both the primary and secondary rovers arrive at the rendezvous point at the times specified by their symbiotic constraints ($t = 10000s$ and $t = 12000s$, respectively). This verifies that the DPC.TF algorithm is able to correctly enforce rendezvous constraints, even if the times are different,



(a) (b) (c)

Figure 4.3: Paths of two rovers with Feeder-Feeder constraints set to different times. In subfigure b, the cyan rover has already made it to its rendezvous point at $t = 10000s$ and continues towards its goal. In subfigure (c), the magenta rover reached its rendezvous constraint at $t = 12000s$, arriving at the same location where the cyan rover was at $t = 10000s$.

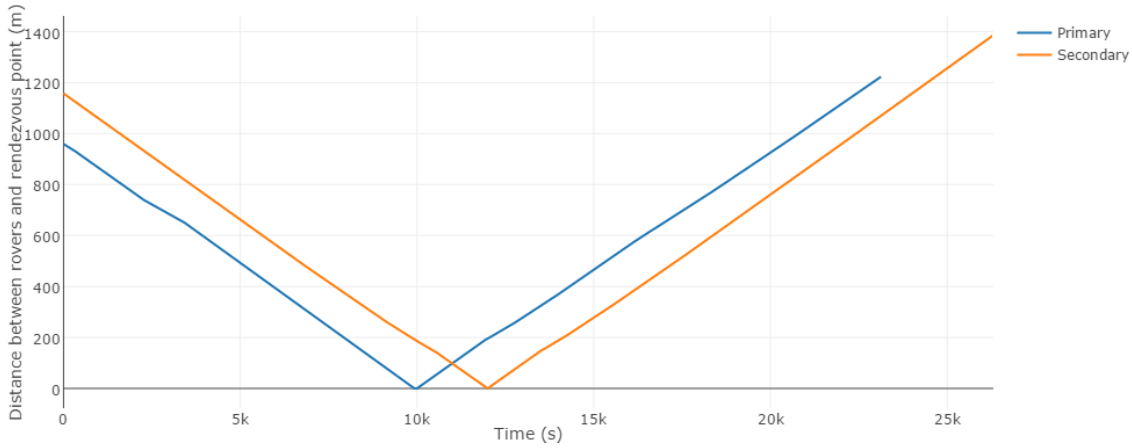


Figure 4.4: Distance between rovers and rendezvous point vs Time

a capability not present in the original algorithm.

4.1.2 Tanker-Feeder

The Tanker-Feeder constraint (where one rover's rendezvous constraint is based upon the state of the other rover) is one of the key constraints within the symbiotic scenario, and is required to meet the condition of resource-based rendezvous. In all scenarios, rovers had identical capabilities unless otherwise noted, and the Tanker rover used time as its determinant (indicating that it would be in the same location as the Feeder when that Feeder's constraint is applied).

Given the complex and non-monotonic nature of the energy model, initial experimentation used distance traveled as the constraint determinant for the Feeder. Specifically, the two rovers should rendezvous after the Feeder has traveled 1300 meters. Figure 4.5 shows the distance traveled by the Feeder and the distance between the two rovers versus time. The DPC.TF algorithm is correctly able to force the Tanker to rendezvous with the Feeder after the Feeder traveled 1300 meters, regardless of the distance traveled by the Tanker rover.

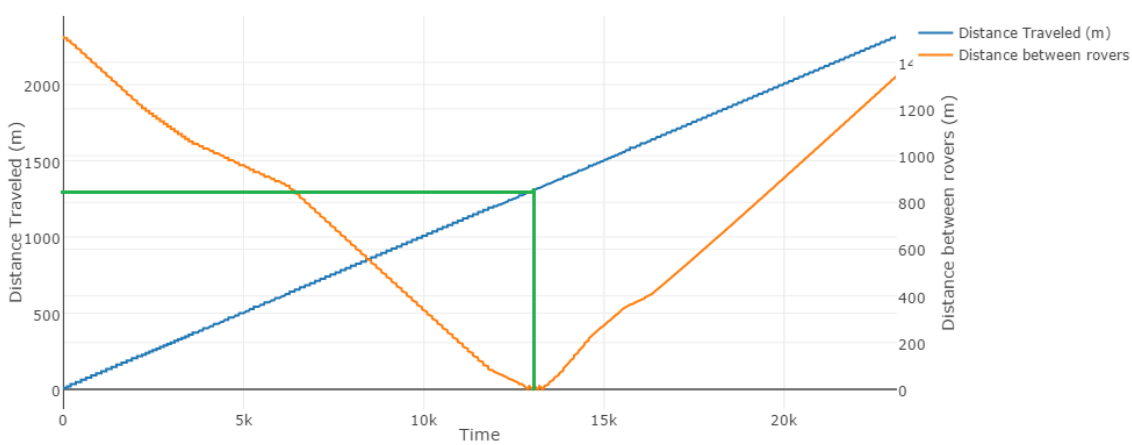
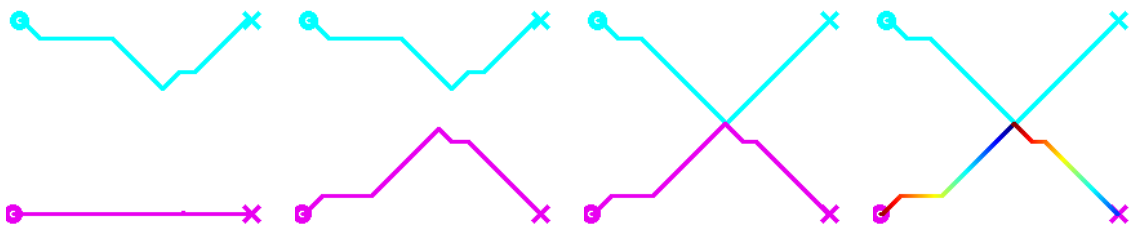


Figure 4.5: Distance between rovers and distance traveled by Feeder vs Time. The green line indicates the time and distance at which the constraint (1300 m) is applied.

Once it was verified that the algorithm operated correctly using distance as the Feeder's constraint determinant, paths using energy as the determinant were evaluated. To do this, some parameter reconfiguration was required. In the previous examples, energy cost of a path was not taken into account; the recharge rate for both rovers was set higher than the maximum power draw. In this scenario, the Feeder's parameters we updated such that it had a 5.5kWh battery, no self-recharging capabilities, a 400W hotel cost, and a 200W drive cost. The symbiotic constraint was specified such that the rover's would rendezvous for battery hot-swapping when the Feeder's battery level reached 3.3kWh. All other parameters remained identical between the two rovers.



(a) Iteration 9 (b) Iteration 10 (c) Iteration 13 (Final) (d) Final paths colored by energy

Figure 4.6: Paths of two rovers with Tanker-Feeder constraint to rendezvous when the Feeder (magenta) reached a battery level of 3.3kWh. Subfigure (d) overlays the energy level for the feeder over the bath, right red indicating high energy and blue indicating low energy.

Figure 4.6 shows the resulting paths when the above Tanker-Feeder constraint is applied. The paths only began to diverge from their initial trajectories after nine DPC.TF iterations, but afterwards they quickly converged to a final solution, show in in Figure 4.6c. In Figure 4.6d, the energy level for the Feeder rover is displayed over the path. Note the decreasing energy level until the rovers successfully rendezvous and the Feeder's battery is swapped. Afterwards, it is able to travel to the goal before its battery runs out, whereas it would have been unable to do so had a rendezvous not occurred. Figure 4.7 shows the energy level of the Feeder and the distance between the two rovers over time. When the Feeder's battery level is at 3.3 kWh, it has rendezvoused with the Tanker and swaps out its battery for a fully charged one.

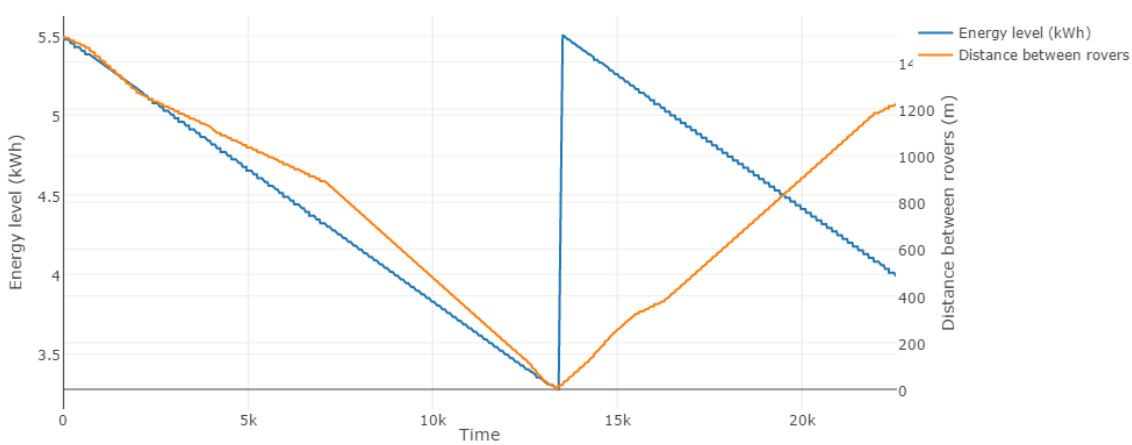


Figure 4.7: Distance between rovers and energy level of Feeder vs time.

4.1.3 Tanker-Tanker

Tanker-Tanker constraints are used to enforce maximum separation distance constraints on the rovers. This is useful for forcing the rovers to remain in communication range of each other for the duration of their path. Unlike previous constraints, the distance portion of the constraint is not zero (indicating a rendezvous), but set to the range of the communications equipment. Because the Tanker-Tanker constraint allows for a much wider range of behaviors, the starting location and goals differ from previous examples. The two rovers start at the same location with multiple goals apiece and must travel to each of the goals such that they do not leave the constraint range.

Results show that the DPC.TF algorithm has a more difficult time enforcing the Tanker-Tanker constraints versus the Tanker-Feeder constraints. It often took many more iterations to converge upon a solution, and the solution it found would oftentimes be nonintuitive, or not fully meet the constraint, even if one was possible. Figure 4.8 shows the distance between the rovers vs time for two robots with a Tanker-Tanker constraint for them to remain within 200 meters and 180 meters of eachother. Between the two experiments, the parameters and goals remained identical. When applying the 200 meter constraint, the robots often fell outside of the 200 meter radius. Yet, interestingly, when a 180 meter maximum range was specified, the robots always were within 200 meters of eachother (and only briefly fell outside of the 180 meter limit. This indicates that a valid path for the 200 meter constraint could have been found with DPC.TF, but was not.

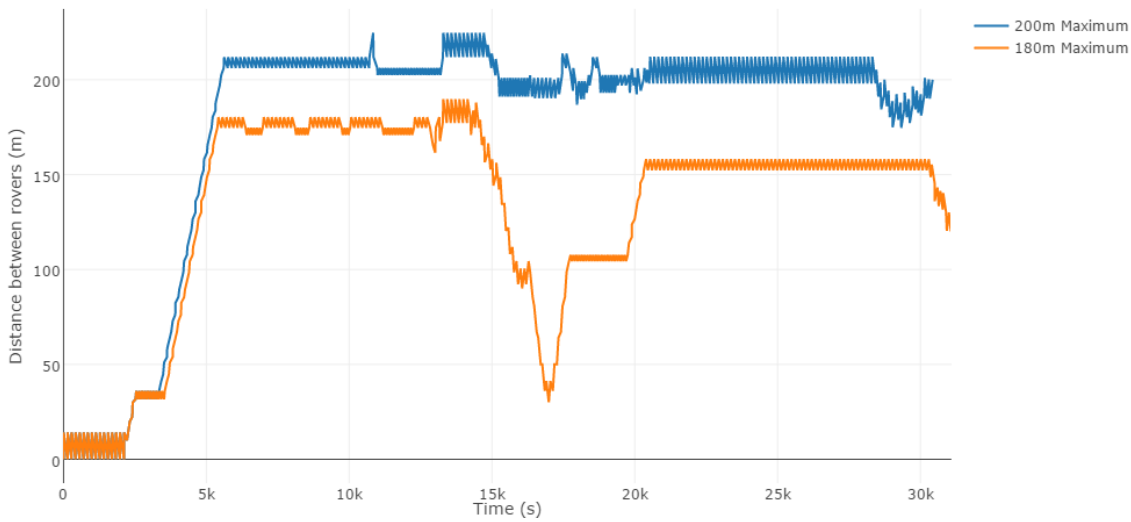
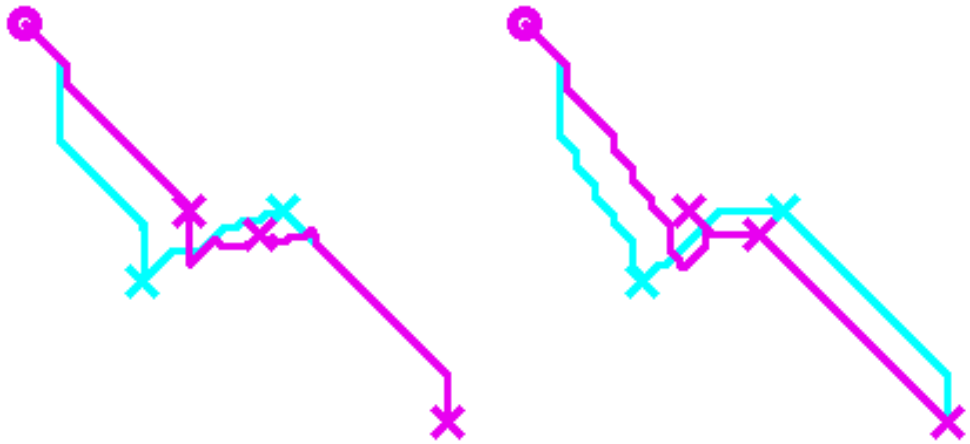


Figure 4.8: Distance between rovers vs Time: Tanker-Tanker

This result is unexpected for a few reasons. First, that not only was planning with the 180m constraint able to find a solution that addresses the 200m constraint, but it was able to mostly meet its constraint while the 200m one did not. Second, given the parameters of the scenario (identical rover capabilities, no limits on energy expenditure or path length), a solution should exist for any maximum distance constraint. In the worst case scenario with a maximum distance 0 meters (indicating the rovers are always right next to eachother), the generated plans should have the rovers follow eachother perfectly to their respective goals. Increasing the distance constraint would still cause such a path to be valid, albeit suboptimal. That they did not find a valid path for the 200m constraint indicates some other condition was interfering with the planning.

Viewing the resulting paths (Figure 4.9) can offer some insight as to what happened. In the case of the 200 meter constraint (Figure 4.9a), the magenta rover headed straight towards its first goal. Looking back at Figure 4.8, it remains within the 200 meter constraint until it begins to



(a) Path with 200m Tanker-Tanker constraint (b) Path with 180m Tanker-Tanker constraint

Figure 4.9: Paths on blank map with Tanker-Tanker constraints

branch away from the cyan rover at around $t = 3500s$. From then on until the goal, the two rovers remain slightly more than 200 meters distance apart from eachother. This is likely because the distance was close enough to the constraint that it was unable to pull the magenta rover away from a direct path to its goal. Once it reaches its first goal, it readjusts its path downwards to meet the constraint instead of directly heading towards its goal. Only after the cyan rover reaches its first goal does the magenta rover continue on towards its second.

Compare this to the paths generated when a 180 meter constraint was specified. In that scenario, the magenta rover does not head directly towards its goal, but more or less matches the path of the cyan rover, with a slight offset to the right. When the cyan rover reaches its goal and continues to its next one (causing it to drive closer to the magenta rover), the magenta heads towards its first goal, remaining within 180 meters of the cyan rover for the duration of its path.

There are a few possibilities for why this discrepancy between the 180 meter and 200 meter constraints exist. The first is that, with the 200 meter constraint, the shortest path to the goal laid just outside the 200 meter range for both robots. Because of this, the penalty was not high enough to cause the rovers to be pulled together. With the 180 meter constraint, the ideal path was well outside the ideal range, leading to a much stronger "pulling force" between the rovers.

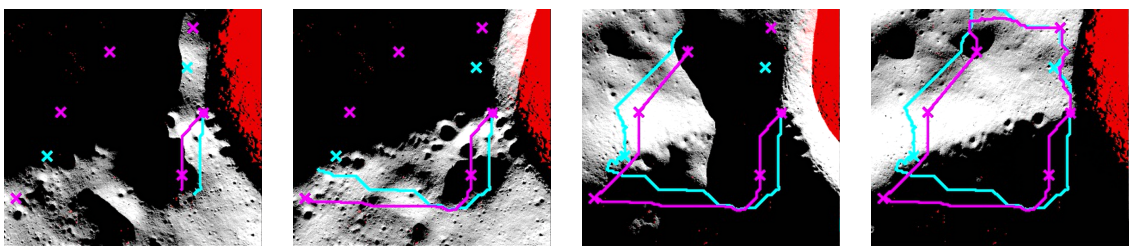
The other possibility stems from the state dominance function used to prune the search space. The state dominance function used to ignore bad states was based upon which state had a lower constraint cost. That is, given multiple states for the same node, the one with the lower DPC.TF error would be favored. If two states had identical constraint costs, the one with the lower time or distance cost (depending upon what cost was being optimized) would be favored. In the case of the 180 meter distance constraint, states which had the rovers 180-200 meters apart would have an associated constraint cost, while the same states with the 200 meter constraint would have a constraint cost of zero. In such instances, where or not a state ignored would be ignored would be dependent entirely upon the time it reached that states corresponding node rather than whether or not it helps satisfy the constraint. Because of this, it is possible that some states were ignored by the state dominance function that would (later in the search) be necessary to produce a valid path.

While the performance of DPC.TF on Tanker-Tanker constraints was inconsistent, a valid path fully meeting the desired maximum distance constraint could still be achieved by lowering the separation distance for the constraint below what was desired. While not ideal, it is a sufficient result to continue experimentation with Tanker-Tanker constraints on Lunar maps.

4.1.4 Lunar Map Performance

4.1.4.1 Tanker-Tanker

The first symbiotic scenario evaluated was one where two rovers must travel to different goals while remaining within 2km of eachother for the duration of this path. To facilitate this scenario, a single Tanker-Tanker constraint was used with a maximum distance of 1.5 km. This value was chosen over 2km to ensure a buffer in case the planner had trouble finding a valid solution for 2km, as discussed in the previous section. Each rovers speed was set to $3\frac{cm}{sec}$ and were allowed to traverse any slope less than 30˚. This present a fairly optimistic scenario in terms of rover capability, but is useful for validating the algorithm on Lunar terrain. Additionally, the primary rover (colored in cyan in future images) was constrained to always remain within the sun. The secondary rover (colored magenta) was allowed to travel through darkened regions, but had a limited battery life. Solar panels recharge the rover when operating in lit regions, but the battery drains when traveling through shadow.



(a) Paths at 50 Hours (b) Paths at 125 Hours (c) Paths at 400 Hours (d) Paths at 500 Hours

Figure 4.10: Paths around Shackleton Crater with a 1.5 km Tanker-Tanker constraint

Figure 4.10 shows the final output of the planner after 20 iterations. Both rovers are successfully able to travel to all their goals while meeting the lighting and energy constraints given to them. Overall, the traversal required 21 Earth days to complete and ended up in the same place as it began. The location it ended up in remains lit until the completion of the Lunar cycle, allowing it to repeat a similar path are the start of the next Lunar cycle. This would allow for a nearly continuous mission scenario.

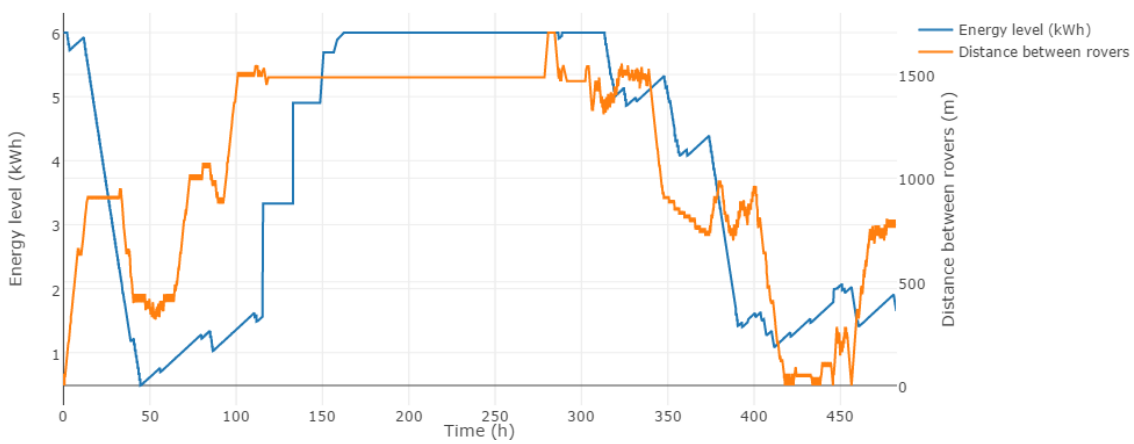
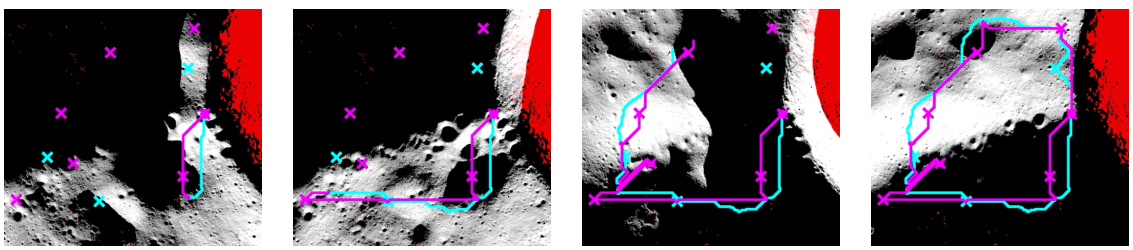


Figure 4.11: Distance between rovers and energy level vs Time: Tanker-Tanker on Shackleton

As seen in Figure 4.11, both rovers remained within the desired 2 kilometers of each other for the duration of the path. As before, the rovers did not meet their supplied constraint of a 1.5 kilometers, leaving that range briefly three times. The maximum distance between the rovers was 1700 meters at 280 hours ($11\frac{2}{3}$ days) into the path. The secondary (magenta) rover also remained above its minimum .5 kWh (battery life for the duration of its path, and spent the majority of its time above 3kWh (50% battery life). These results indicate that a valid path for the given constraints does exist and the route would be a feasible one with the given parameters.

4.1.4.2 Tanker-Feeder

The above configuration assumed that the secondary rover was capable of self-charging. In some symbiotic configuration, such a capability would not be possible, and it must instead rely upon the primary rover to either recharge or swap out its batteries. Such a limitation requires a Tanker-Feeder constraint to be applied. Additionally, without the added load of solar panels, the rover may be able to travel faster. In the below scenario, the velocity for the primary rover remained the same, while the velocity of the secondary rover was set $1\frac{cm}{sec}$ higher to $4\frac{cm}{sec}$.



(a) Paths at 50 Hours (b) Paths at 125 Hours (c) Paths at 375 Hours (d) Paths at 500 Hours

Figure 4.12: Paths around Shackleton Crater with a Tanker-Feeder constraint

Figure 4.12 shows the resulting path using this configuration. A solution to the energy constraints was found after 45 iterations. The rovers rendezvoused a total of nine times during the path and, as before, arrived in the same location the started from. The distance between rovers went above 2km twice, topping out at 2.4km. This is still far less than the maximum range of most

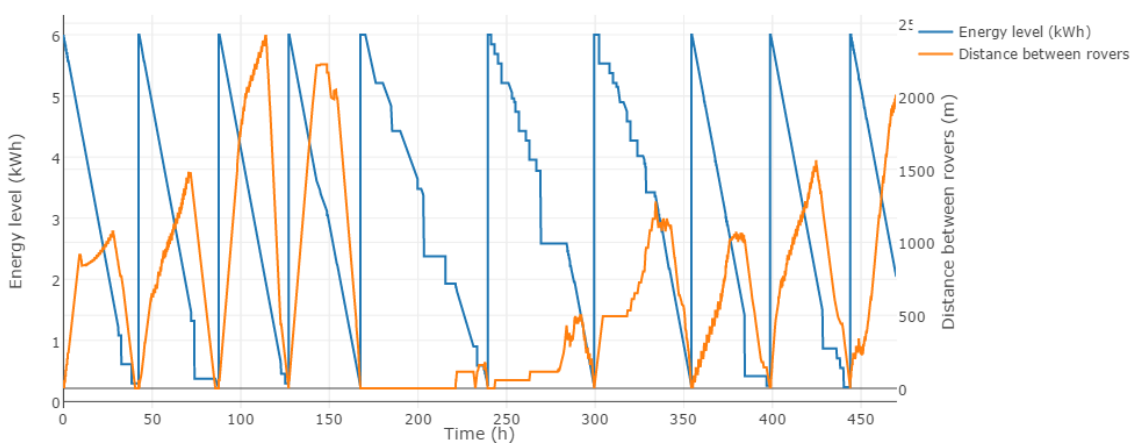


Figure 4.13: Distance between rovers and energy level vs Time: Tanker-Feeder on Shackleton

high-powered short-range radios. Interestingly, both the path duration and total distance traveled were almost identical between the Tanker-Tanker and Tanker-Feeder experiments.

4.2 Parameter Effects

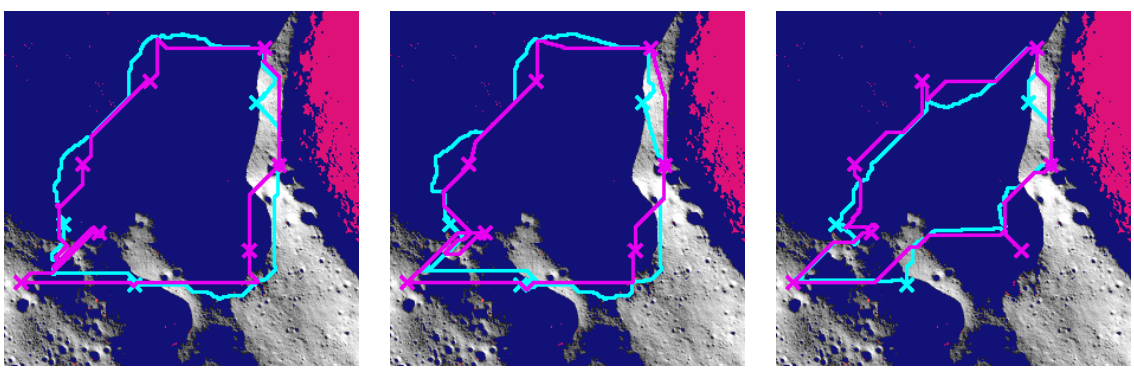
The parameters used to generate the above paths are realistic, they are highly optimistic. Developing a rover with such capabilities may prove to be more costly than necessary, especially if acceptable routes can be found with more common parameter configurations. This section shows how different configurations affect the planner output, and whether some configurations are not possible with the given constraints.

4.2.1 Robot Capability

4.2.1.1 Velocity

While $3\frac{cm}{sec}$ and $4\frac{cm}{sec}$ for the primary and secondary rovers, respectively, is certainly possible, they are very high speed compared to the 1-2$\frac{cm}{sec}$ speed most rovers achieve. To test the limits of what could be achieved, the planner was rerun assuming both rovers had identical velocities of 1, 2, and 3 $\frac{cm}{sec}$.

Figure 4.14 shows the plans for the configurations where a valid route could be calculated. No viable trajectory could be found for the $1\frac{cm}{sec}$ velocity configuration; the first goal was too far in the dark to reach before the battery of the secondary rover ran out. The configuration where both rovers could travel up to $3\frac{cm}{sec}$ is remarkably similar to the baseline, not only in duration and distance traveled, but in the overall shape of the trajectory. This makes sense, however, given the rendezvous constraint that was applied. Because the secondary rover required frequent rendezvous' with the primary rover, it could not stray too far away from it.



(a) Baseline - $3\frac{cm}{sec}/4\frac{cm}{sec}$ (b) Both $3\frac{cm}{sec}$ (c) Both $2\frac{cm}{sec}$

Figure 4.14: Final routes of rovers with different velocity configurations.

The configuration where both rovers has a top speed of $2\frac{cm}{sec}$, however, differs significantly from the baseline. The lower speed meant that routes that were once well-lit for the rovers would be shadowed by the time they arrived, necessitating a different route. This effect can be clearly seen with the first goal in the bottom right of the map. Where as both the baseline and $3\frac{cm}{sec}$ configurations had the rovers travel below the goal, in the $2\frac{cm}{sec}$ configuration they were required to travel above it through a tight ridgeline. The same effect is seen with the center-top goal. Initially, the rovers traveled up and around the crater where the goal lies, but in the lower velocity

configuration, they has to travel below it. As a whole, the route generated at the lower velocities is much tighter than those with rovers capable of traveling at higher speeds.

The lower $2\frac{cm}{sec}$ model has an additional, beneficial, side effect over the other configuration. Because the velocities were so low, the rovers remained within a short distance of each other for the duration of their paths. Figure 4.15 shows the distances between the two rovers over time. With the exception of a brief moment in the middle of the path where the rovers where 1.2km apart, the rovers remained within 1km for the entire route. Such a result precludes the need for a dedicated Tanker-Tanker constraint to enforce a communication range constraint: the rendezvous constraint essentially builds it into the plan.

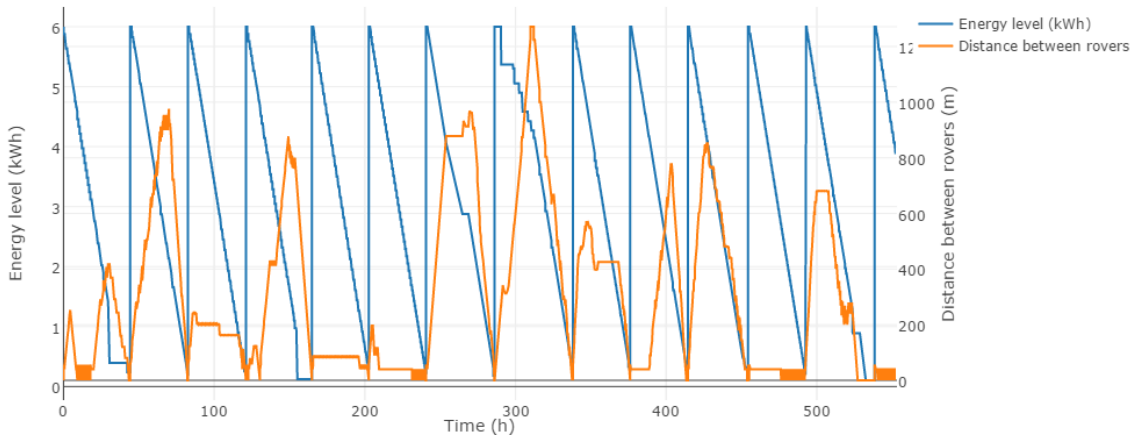
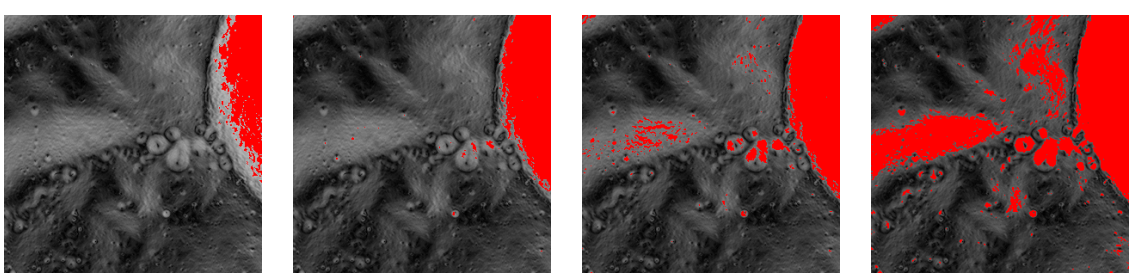


Figure 4.15: Distance between rovers and energy level vs Time: $2\frac{cm}{sec}$ configuration on Shackleton

4.2.1.2 Hazards

Besides just velocity, terrain capability is another constraint that must be tuned for each rover. Some rovers will be more capable of traversing rough or steep terrain than others. For these experiments, the parameters displayed in the $2\frac{cm}{sec}$ configuration from above are used as the baseline for comparison, having both rovers capable of traversing 30˚ slopes. The experiments conducted varied the maximum slope between $15^\circ - 30^\circ$. Figure 4.16 shows slope maps of Shackleton Crater, with regions above these thresholds highlighted in red.



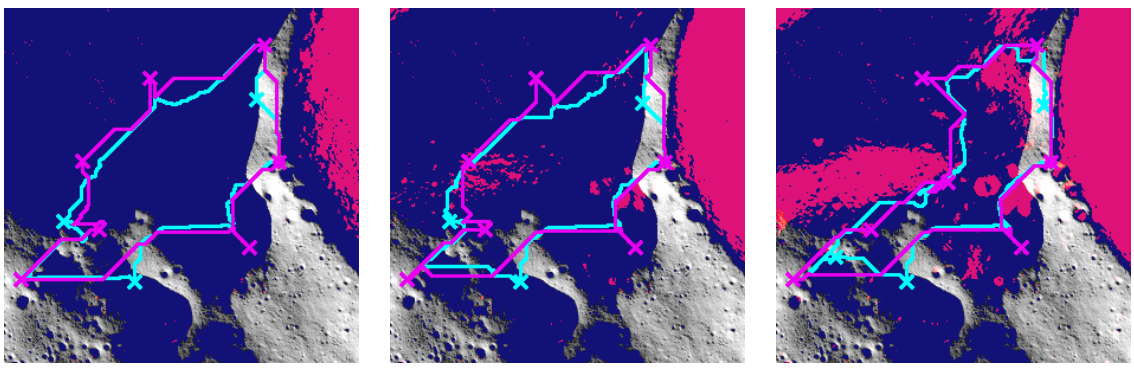
(a) Slope > 30˚ (b) Slope > 25˚ (c) Slope > 20˚ (d) Slope > 15˚

Figure 4.16: Slope maps of Shackleton crater with slope thresholds. Red indicated the area contains slopes greater than or equal to the corresponding threshold.

Figure 4.17 shows the final trajectories for each configuration. The paths did not vary sig-

nificantly between the the 30˚/30˚ and 20˚/25˚ configurations. The path durations and distance where nearly identical (23 days total and roughly 32km per rover, respectively). Indeed, the only significant difference between the two was a slight different route taken down the slide of the major hill towards the left of the map. The symbiotic constrains were still able to be met throughout the path, although this slight detour did increase the maximum distance between the rovers to 1.4km.

The paths did, however, vary significantly when the maximum slopes for the primary and secondary rovers was set to 15˚ and 20˚, respectively. Although the large hilly region towards the left of the map was mostly traversable for the secondary rover, it was completely off-limits for the primary rover. As the velocity of the rovers was such that a rendezvous would have to occur within that region, an alternative route around the hilly area was taken. This resulted in a much longer duration path, as the rovers had to wait for it to be lit (so the primary rover could travel through it). This added approximately three days to the overall mission duration, and an additional 2km of travel for each rover. A valid path was still found meeting all constraints, and the rover remained within 1km of eachother for the vast majority of the path and within 1.3km of eachother for the entirety of the path.



(a) Baseline - 30˚/30˚ (b) 20˚/25˚ (c) 15˚/20˚

Figure 4.17: Final routes of rovers with different hazard configurations.

4.2.2 Energy Model

The baseline energy model assumed a 6kWh battery, a hotel cost of 80W, a drive cost of 75W, and a rover velocity of $2\frac{cm}{sec}$. Based upon the analysis for Lunar rover power requirements presented in [65], these are reasonable values for a less-featured rover. Still, they do not allows for much leeway in terms of rover capability. A 6kWh battery may not be feasible for a specific mission, requiring that the capacity be lower. On the otherhand, if more power is available, a more capable rover could be designed. These questions necessitate multiple power configuration that need to be evaluated. The configurations that were run are as follows:

A) Baseline - Velocity: $2\frac{cm}{sec}$, Battery: 6kWh, Hotel Cost: 80W, Drive Cost: 75W

B) Velocity: $2\frac{cm}{sec}$, Battery: 5kWh, Hotel Cost: 80W, Drive Cost: 75W

C) Velocity: $3\frac{cm}{sec}$, Battery: 5kWh, Hotel Cost: 80W, Drive Cost: 75W

D) Velocity: $1.5\frac{cm}{sec}$, Battery: 7kWh, Hotel Cost: 80W, Drive Cost: 75W

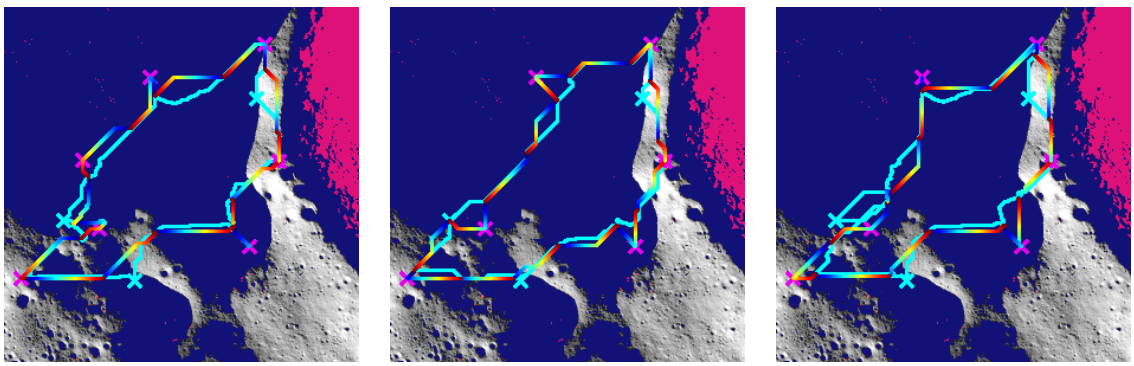
E) Velocity: $2\frac{cm}{sec}$, Battery: 7kWh, Hotel Cost: 100W, Drive Cost: 75W

F) Velocity: $2\frac{cm}{sec}$, Battery: 7kWh, Hotel Cost: 100W, Drive Cost: 100W

G) Velocity: $2\frac{cm}{sec}$, Battery: 6kWh, Hotel Cost: 100W, Drive Cost: 75W

The battery capacity of the rover was the first property that was modified. Initial experiments lowered the capacity to 5kWh (configurations B and C). It was found that, at this battery capacity, a path that met the rendezvous constraints could not be found, even when the velocity of the rover was increased. This suggests, unsurprisingly, that battery capacity plays a major role in determining whether or not a valid route exists.

To validate this hypothesis, experiments were run using a 7kWh battery capacity configuration, and varying other parameters, such as velocity, hotel cost, and drive cost (configurations D-F). Figure 4.18 shows the paths with these different configurations relative to the baseline. Valid routes were found for the configurations D and E, but could not be found for configuration F. An additional experiment was done using the baseline 6kWh battery capacity, but a 100W hotel cost. The planner was not able to find a valid route using those parameters. These results indicate that the hypothesis that battery capacity plays a significant role on whether or not a valid route can be found is correct.



(a) Baseline Configuration (A) (b) Configuration (D) (c) Configuration (E)

Figure 4.18: Paths with energy levels for different energy model configurations. The Feeder rover's energy level along the route is overlayed on the path, with red indicating high energy and blue indicating low energy.

4.3 Optimizing Functions

4.3.1 Goal Heuristics

To validate the performance of the heuristics, the Chebyshev, Euclidean, and Octile distance heuristics, as will as a time-based heuristic were evaluated. A better performing heuristic should get result in fewer states visited and expanded, and should ideally require less processing time total. Additionally, it should converge upon a solution in fewer iterations. In the below results, the results were calculated using two robots with a single Feeder-Feeder rendezvous constraint set to occur after both robots traveled 1000 meters. All settings were identical between the two rovers, and the only settings that varied across the experiments were the goal heuristic and the DPC penalty increase rate (ϵ).

Though these experiments were primarily based around evaluating the goal heuristic, the DPC ϵ required changing to ensure like conditions were being evaluated. The reason for this stems from how the discretization of the search space and the units used. In the blank world, the smallest distance step is 10 meters, and the smallest time step is 400 seconds. As both rovers were given a top speed of $.1\frac{cm}{sec}$, this means that the f-cost of a path that is optimizing for time will generally be higher than the f-cost of a path optimizing for distance. The DPC penalty is always based upon

the distance between two rovers. Therefore, when optimizing for time, a higher weighting must be applied to the DPC penalty in order to overpower the higher value of the heuristic cost.

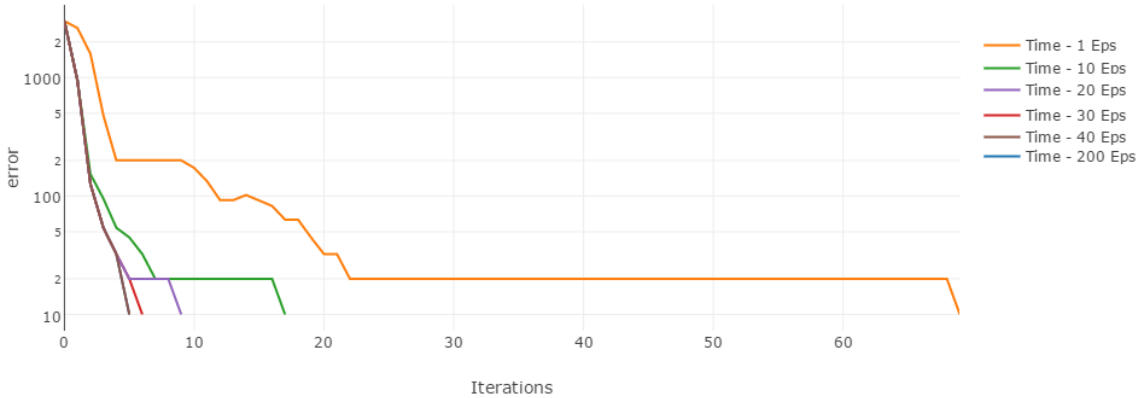
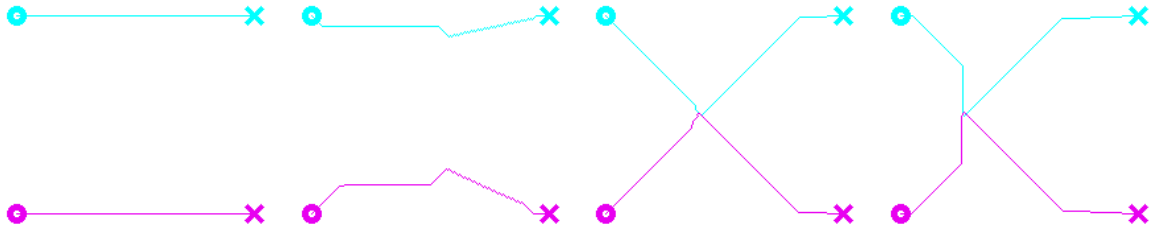
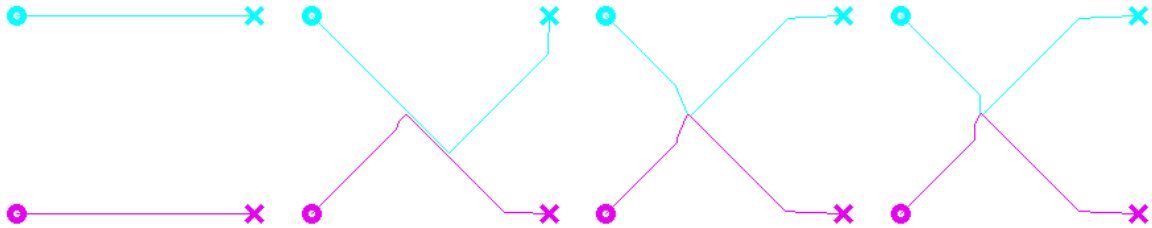


Figure 4.19: Constraint error vs DPC iteration

This effect can be seen in Figure 4.19. Note, that as the penalty epsilon increases,the number of iterations required to converge to a solution decreases. The convergence bottoms out after $\epsilon = 40$. This is because, as stated before, a lower ϵ means a lower DPC penalty is applied, meaning the rover will not be drawn too far away from the optimal path to the goal. Figure 4.20 shows the convergence process using an ϵ of 1 and 40. Note that it takes slightly more iterations for $\epsilon = 1$ to come to a close solution and significantly longer for it to converge towards an exact solution when compared to $\epsilon = 40$.



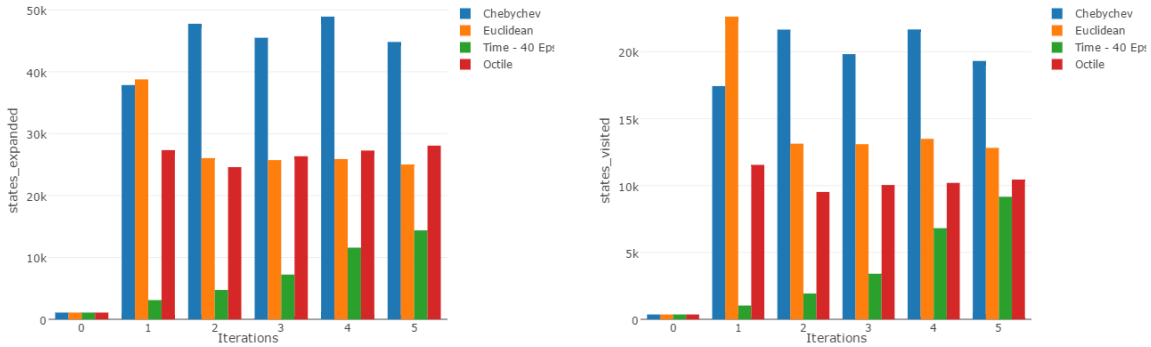
(a) Eps 1 - Init (b) Eps 1 - Iter 1 (c) Eps 1 - Iter 10 (d) Eps 1 - Iter 60



(e) Eps 40 - Init (f) Eps 40 - Iter 1 (g) Eps 40 - Iter 3 (h) Eps 40 - Iter 5

Figure 4.20: The path of each rover at different DPC iterations with time heuristics. Figures a-d show the convergence sequence with $\epsilon = 1$, while Figures e-h show the convergence sequence with $\epsilon = 40$.

Now that an acceptable ϵ has been chosen for time, it can be compared to the distance heuristics. Figures 4.21 shows the number of states expanded and visited for each heuristic by DPC iteration. As expected, the Chebychev heuristic (which severely underestimates the distance to the goal) performs much worse than the other three. The performance of the Euclidean and Octile heuristics, however is much more interesting. As expected, the number of states visited by the planner is much lower when using the Octile heuristic versus the Euclidean heuristic. As explained in Section 3.2.2.4, this is because the Octile heuristic acts nearly perfectly on an 8-grid. As neighbors are expanded at each state visitation, it would be expected that the Octile heuristic would have comparably less estates expanded. As Figure 4.21a shows, this is not the case, and the Euclidean heuristic either has a similar or slightly lower number of states expanded at each iteration (other than the first).



(a) States Expanded (b) States Visited

Figure 4.21: The number of states expanded and visited by each heuristic at each iteration.

Table 4.1 (which shows performance statistics across all iterations) reveals the cause behind this discrepancy. The Euclidean heuristic visits more states yet expands fewer than the Octile heuristic because the state dominance function causes it to ignore roughly 50% more states than the Octile heuristic. Recall the process of the state dominance function. When a state is visited, a state dominance function is called to evaluate whether or not it should be expanded. If the state does not dominate any others, it will not be expanded (ie. it is ignored). With the Euclidean heuristic, many states are processed multiple times. Of these, only a few pass the state dominance function and are added to the priority queue (expanded).

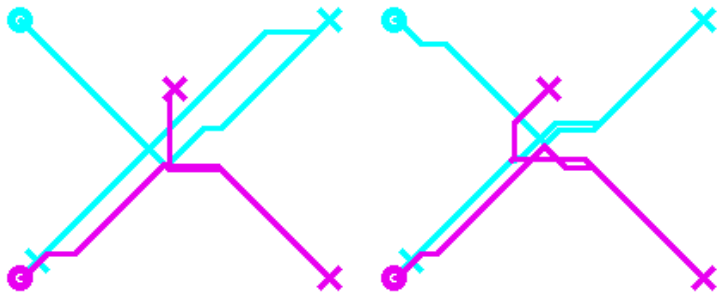
States:	Expanded	Visited	Removed	Ignored
Chebyshev	225955	100261	131190	576133
Euclidean	130106	69151	65742	422955
Time - 40 Eps	35718	18786	18849	114570
Octile	134718	52111	74998	281999

Table 4.1: Performance of different heuristics across all iterations.

Also of note is how well the time-based heuristic performed on all metrics. In every case, it visited, expanded, and searched through significantly fewer states than all others. Interestingly, as Figure 4.21 shows, the numbers consistently increase at each iteration, compared to the other heuristics which stay more or less consistent across iterations.

The choice of heuristics and objective function also affects how the symbiotic constraints are met. Assume a Feeder-Feeder constraint where both rovers are to rendezvous at $t = 12000s$ and $t = 39000s$. Figure 4.22 shows the path resulting from this constraint when distance (a) and time

(b) are the costs being minimized and multiple goals are present. While both paths meet the given constraints, the results are very different.



(a) Optimized for Distance (b) Optimized for Time

Figure 4.22: The paths generated by DPC.TF for Feeder-Feeder constraints at $t = 12000s$ and $t = 39000s$ with planners optimizing for distance and time.

Figure 4.23 shows the reason for this discrepancy. In the distance optimized path, the cyan (Primary) rover does not move after its initial rendezvous. Instead, it remains in the same location and waits for its partner, the magenta (Secondary) rover, to travel to its goal and return to it at $t = 39000s$. The secondary rover actually ends up arriving early, and waits in the same position as the primary rover until $t = 39000s$, after which the both continue on towards their goals. This process, while time consuming, results in the least amount of distance traveled by both rovers, hence why it is chosen. But having a rover sit and wait for extended periods of time is highly inefficient.

When optimizing for time, the actions are totally different. Immediately after rendezvousing, the rovers continue on towards their goals. Their paths are once again diverted to the second rendezvous constraint at $t = 39000s$. The rovers are moving for almost the entire duration of the path; there is little to no waiting because a longer distance path is preferred over a longer duration

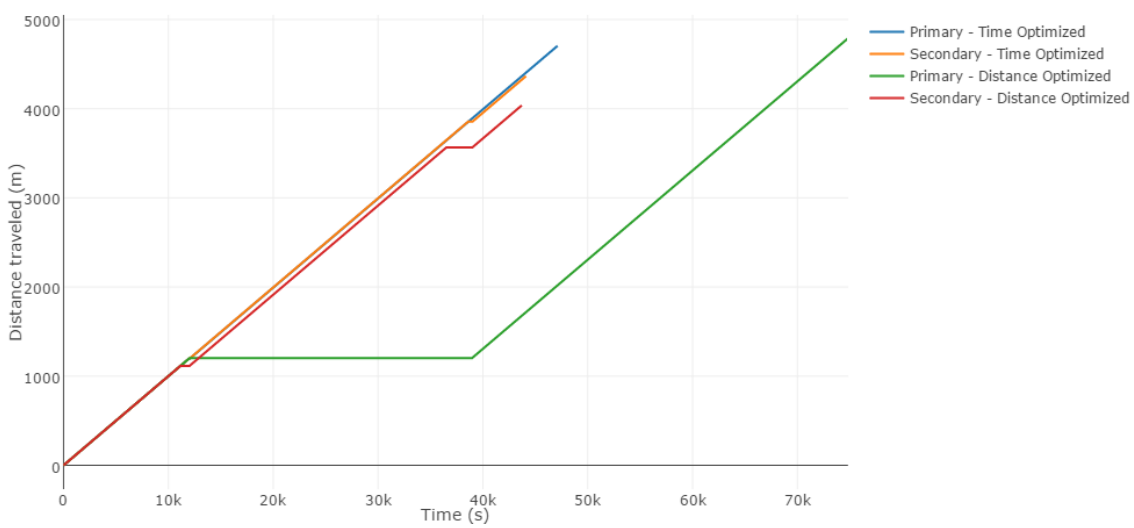


Figure 4.23: Distance traveled (m) vs Time for rovers with Feeder-Feeder constraints at $t = 12000s$ and $t = 39000s$.

path. While the paths may be longer, the result is a much more efficient path in terms of time usage, as there is almost no time spent waiting.

4.3.2 Forward Cost Projection

The purpose of implementing Forward Cost Projection is to limit the amount of states that need to be searched before one that minimizes the symbiotic constraint is found by guiding the planner towards the location that minimizes the constraint earlier on in the search. As Tanker constraints are essentially always active and applied, only constraint pairs involving a Feeder see benefit from FCP. For this reason, validation was conducted using Feeder-Feeder constraints in the open world.

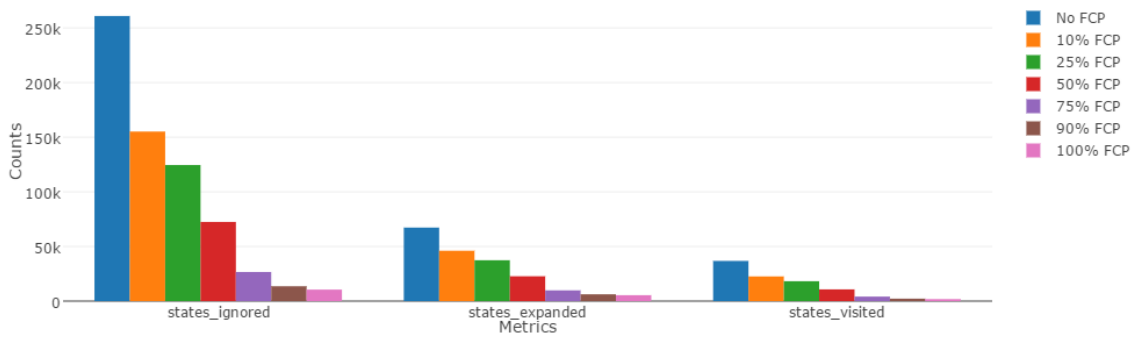
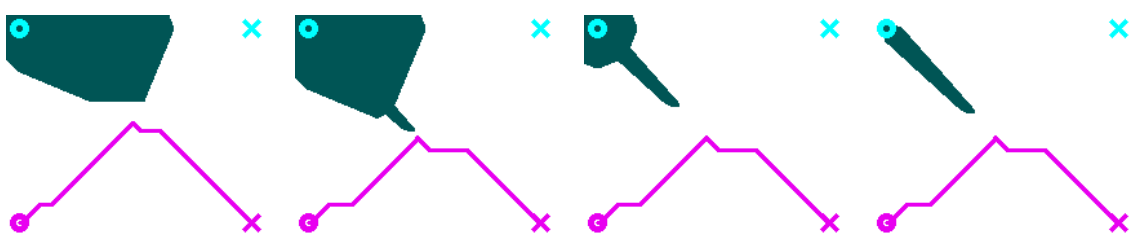


Figure 4.24: Performance of Feeder-Feeder constraint with different FCP thresholds.

For validation, a Feeder-Feeder constraint to rendezvous at $t = 12000s$ was given to both robots (similar to the one described earlier in Figure 4.1. Thresholds for the FCP constraint to occur were set between 0-100%, with 0% indicating the penalty would only apply exactly when the determinant cost for state was equal to the constraints value, 50% indicating it would apply when halfway to that value, and 100% indicating that a penalty for the constraint would be computed at all times. Figure4.24 shows the performance of the constraint with different thresholds at which the constraint is applied. As expected, applying the constraint earlier leads to fewer states that are both expanded and visited. Fewer states are ignored simply because fewer states are evaluated at higher FCP thresholds. Of those states that are evaluated, whoever, more are ignored at lower thresholds.

This effect is seen in Figure 4.25, which shows the states explored at a snapshot during planning at different FCP thresholds. When FCP is not applied, the planner initially plans directly towards



(a) No FCP (b) 25% FCP (c) 75% FCP (d) 100% FCP

Figure 4.25: States expanded at various FCP thresholds. Darkened pixels indicate nodes that have been visited or expanded during path search.

the goal, only expanding its search outwards once the constraint is applied. When it does expand outward, it searches using uniform cost search. When FCP is applied, the planner will be drawn towards the intended rendezvous point earlier on in the plan. How early depends upon the FCP threshold.

States:	Expanded	Visited	Removed	Ignored	Time (ms)
No FCP	221807	116856	25114	822788	2753.177
10 % FCP	166768	81740	927	563192	1753.099
25% FCP	135801	66169	1197	454624	1498.077
50% FCP	88374	41922	2272	285163	1038.071
75% FCP	51664	22903	3613	152571	690.031
90% FCP	42345	17956	4140	118235	598.031
100% FCP	40800	17095	4314	112184	597.032

Table 4.2: Performance statistics for different FCP thresholds across all iterations

Initially, it was thought that an FCP threshold of 100% may not be ideal because (while less states are explored), the penalty calculations must be applied at every expansion, rather than a handful of them. The metrics shown in Table 4.2, indicate that this intuition was incorrect. Forward projecting the constraint penalty at every state yields benefits over partial FCP in every metric, including total exploration time.

Chapter 5

Conclusion

5.1 Key Findings

The results presented in the previous section lead to a number of key findings. First and foremost, the DPC.TF algorithm presented is capable of generating plans that address the symbiotic constraints of resource based rendezvous while maintaining a maximum separation distance for inter-rover communication. This is a challenge that existing algorithms were unable to easily tackle. Furthermore, the algorithm is able to represent highly complex constraints in a simple and intuitive manner.

The algorithm was also able to show that viable routes following the symbiotic constraints exists on prospective areas of interest on Shackleton Crater. The symbiotic architecture enables rovers to explore permanently shadowed regions that would otherwise be considered off-limits to rovers. These areas present a massive increase in scientific value over areas that can currently be reached by only staying within "safe" regions of the Moon. Additionally, because the discovered routes all returned to their starting location within a single Lunar cycle, the opportunity exists for vastly extended mission duration, as the routes can more or less be repeated for multiple Lunar cycles.

A great deal was learned about the DPC.TF algorithm through its implementation as well. First while Feeder-Feeder and Tanker-Feeder constraints will almost always converge upon an error-free solution, Tanker-Tanker constraints tend to have trouble consistently meeting their assigned maximum distance. This error is likely related to the implementation of the algorithm, rather than the algorithm itself. Specifically, the state dominance function used to prune weak or duplicative states was removing states that were actually necessary for a plan to be found. Fortunately, the error is almost always small, and easily addressable by setting the value for the Tanker-Tanker constraint to be slightly less than the desired maximum distance.

Incidentally, a tight Tanker-Feeder constraint that required relatively frequent rendezvous between rovers was often enough to enforce the maximum separation distance constraint on its own, depending upon the capabilities of the rover. If the velocities of the rovers were on the lower end, the rovers would essentially be forced to remain within communication distance if they were to be capable of rendezvousing by the time the energy constraint was met.

Significant work was also done to find how best to optimize the entire planning process. Time based heuristics were better than their distance based counterparts at finding viable paths. While this result means that the cost being minimized is path duration, rather than path distance, the results were often far superior and faster to calculate. Distance based heuristics would result in paths that would have the rovers sitting and waiting for extended periods of time that could have been spent exploring.

Forward Cost Projection was also found to be extremely beneficial for quickly finding paths towards the rendezvous constraints. Though less useful for Tanker-Tanker constraints, Feeder-

Feeder and Tanker-Feeder constraints saw a significant reduction in the number of states that needed to be explored before finding a solution to the rendezvous constraint without any performance penalty.

Finally, while a state dominance function is absolutely essential to pruning the massive search space, great care must be taken to ensure that it does not negatively affect the performance of DPC.TF. The chosen state dominance algorithm that sought to minimize constraint cost, time, and energy (in that order) worked well in most situations, but caused problems when dealing with Tanker-Tanker constraints.

Overall, the DPC.TF algorithm and the resulting planner were successful in achieving their stated goals. The constraints worked almost exactly as expected and the results generated by applying the planner to maps generated from actual Lunar data are a promising starting point for future research into Symbiotic Exploration.

5.2 Future Work

The DPC.TF algorithm as presented is a solid basis for addressing constraints within symbiotic planning, and was able to show that valid routes to high-interest locations exist while meeting symbiotic constraints. Yet, it is presented more as a proof-of-concept than an actual implementation that should be used for future missions. There are many capabilities that it is missing that would further enhance its capability both within the context of symbiotic exploration and outside of it. Some of these involve improvements to the DPC.TF algorithm itself, while others target the greater capabilities of the developed planner. Finally, this research can be expanded beyond what was discussed into other prospective missions sites, both on the Moon, Mars, and on Earth as well.

5.2.1 DPC.TF Improvements

Range Constraints One capability that DPC.TF in its current form is missing is the ability to specify ranges of values for a constraint to apply at. Each Feeder constraint only has a single value at which it is applied. However, it may not be the case that rendezvous must occur when an exact condition is met. For instance, rather than rendezvousing when the Feeder is at exactly 10% battery life, a more useful constraint would have it rendezvous when the battery life is somewhere between 10% and 30%. There is currently no way to specify such a constraint. This is problematic, as giving too high a value for the constraint will cause frequent rendezvous that may detract from exploration potential, while specifying too low a value either makes the resulting plan riskier (as there is less room for error), or will cause the robots to rendezvous early and simply wait together for the constraint condition to be met.

Durational Rendezvous Currently, all rendezvous are instantaneous in nature; as soon as they occur, the robots continue towards their goals. While instantaneous rendezvous would be applicable for short-duration actions such as swapping a battery or dropping off samples, more time consuming actions, such as recharging a battery, require that the rovers be together for an extended duration. One potential method of enabling this capability is to store within each state whether or not a rendezvous has occurred and modify the IS_TRIGGERED function within Algorithm 3 to check how long the rover has been in a rendezvous state. If the previous state was a rendezvous state and the rover had not been in a rendezvous state for the specified amount of time, the constraint would be triggered regardless of the determinant cost of the given state.

Multi-Partner Constraints Each type of constraint only has one pair of rovers associated with it. Constraints that must apply between each possible pair of multiple rovers can be easily specified by defining multiple constraints, but there is no way to specify an OR style constraint, where the

constraint can be applied with any rover from a list. Such a scenario would be useful in situations where there is more than one Tanker rover that can supply a Feeder with energy. Such a capability would also enable Tanker-Tanker constraints to operate as a communications mesh where each rover must remain within communications range of at least one other rover. Adding multiple possible partner rovers to each constraint and picking the closest distance partner when evaluating the penalty to apply should be explored as a method of addressing this capability.

DPC Super-Iterations The original DPC paper noted that, in obstacle filled spaces, DPC may either find suboptimal solutions or fail to find solutions at all if the initial path and the ideal path are not homotopic [53]. Their approach was to run the original DPC approach multiple times in "super-iterations" if a valid solution could not be found using basic DPC. Throughout these super-iterations, a blacklist of joint state-spaces between robots is maintained for constraints that were not met. Joint states within this blacklist are avoided in future super-iterations, eventually leading to the initial plan being within the same homotopic class as the ideal plan. Given that DPC.TF suffered from similar issues when dealing with complex Lunar environments, it is worthwhile to see if introducing super-iterations can improve performance. The addition of super-iterations should also help address the challenges Tanker-Tanker constraints had interacting with the state dominance function.

5.2.2 Planner Improvements

Task Allocation In the current implementation of the planner, waypoints are manually assigned and ordered for each robot. Ideally, the rovers would be able to decide amongst themselves which goal each should go to (and which should be skipped altogether) and what order they should travel to them. Unfortunately, this is a variant of the orienteering problem and is NP-Hard, making the challenge of finding an optimal solution intractable when dealing with many waypoints. The iterative nature of DPC.TF compounds this problem. Several algorithms exist for quickly finding acceptable solutions, such as simulated annealing, any-colony optimization, and genetic algorithms. These approaches should be evaluated to see whether or not better routes to the existing goals and others exist.

Improved Robot Model The current robot model used by the planner is, in many ways, too simple for effective use in realistic scenarios. The energy model does not take into account the strength of the solar flux in a location when determining how much solar power should be supplied, just whether or not it exists. Although this was most due to limitations of the lighting maps, a more accurate energy model would assist in accurate path generation. How slopes are handled must also be improved. Each node has a single slope value indicating the maximum slope on that node in any direction. Similarly, each rover has a single limit to how steep a slope in can traverse in any direction. Realistically, rovers can travel down much steeper slopes than they can travel up, and can traverse sideways across many steep slopes. An improved hazard model for the robot should take into account the direction of slope and travel when determining its cost.

Additionally, the model used for communication constraints was not entirely accurate. While maintaining a maximum separation distance is important to maintaining communication between rovers, there are other effects that play an important role in determining whether or not a stable communications link between two rovers can exist, such as line-of-sight. Fortunately, integrating line-of-sight into the calculation of communications constraints is not difficult. Rather than use the Euclidean distance between the two rovers as the basis of the separation constraint, an "effective distance" function could be used. This function would weight the distance between the two rovers based upon the number of cells between them occupied by other barriers (such as hills or mountains) and use the output as the effective distance between the two rovers.

5.2.3 Mission Scope

Additional Lunar Sites Due to its high interest and high fidelity DEM data, Shackleton Crater was the only prospective Lunar mission site thoroughly explored by this research. Plenty of other Lunar regions along the poles are of high interest. These sites include from Haworth crater (which has some of the lowest average temperatures of any region near the South pole [63]), Nobile Crater, and Malapert Peak. The developed planner should be run through all of these areas of interest to determine what benefit a symbiotic architecture could yield.

Non-Lunar Planetary Exploration Besides just Lunar scenarios, Symbiotic Exploration has the potential to yield significant value in other planetary missions. Among the most promising of these is the search for evidence of life along Martian Recurring Slope Lineae (RSL). RSL are dark markings along the surface of steep slopes on Mars thought to be caused by seasonal flows of briney water [66]. As such, they offer one of the best chances of finding evidence of life on Mars. However, the steep slopes and the risk of contamination precludes near-site landing as well as the use of ground rovers for exploring the RSL. Instead, aerial rovers have been proposed as an alternative to explore these valuable features. A symbiotic architecture would be ideal for such a scenario, as a mobile, ground-based rover could carry multiple aerial rovers close to the RSL site before launching them. In such a situation, it is highly likely that the aerial rovers must maintain communication distance with the ground rover, as well as occasionally rendezvous with it for recharging.

Terrestrial Symbiotic Planning While the heavy focus of Symbiotic Exploration has been towards planetary exploration, the model could be just as useful in other contexts on Earth. While the purpose of such a scenario may not necessarily be to reduce risk, it would be useful for rovers traveling between waypoints while remaining within communication for each other. These scenarios could be similar to the ones presented earlier in the background section with the added benefit that precomputed trajectories are not required and rendezvous at fixed time intervals could be forced.

Bibliography

[1] J A George, G W Mattes, K N Rogers, D F Magruder, A J Paz, H M Vaccaro, R S Baird, J T Smith, J W Quinn, W E Larson, A Colaprete, R C Elphic, T R Suaris, and Moffet Field. Resolve Mission Architecture for Lunar Resource Prospecting and Utilization. In *43rd Lunar and Planetary Science Conference*, pages 4–5, 2012.

[2] Bill Steigerwald. Lunar Pits Could Shelter Astronauts, Reveal Details of How 'Man in the Moon' Formed, June 2014.

[3] Issa a.D. Nesnas, Pablo Abad-Manterola, Jeffrey a. Edlund, and Joel W. Burdick. Axel Mobility Platform for Steep Terrain Excursions and Sampling on Planetary Surfaces. *2008 IEEE Aerospace Conference*, pages 1–11, March 2008.

[4] Issa a.D. Nesnas, Jaret B. Matthews, Pablo Abad-Manterola, Joel W. Burdick, Jeffrey a. Edlund, Jack C. Morrison, Robert D. Peters, Melissa M. Tanner, Robert N. Miyake, Benjamin S. Solish, and Robert C. Anderson. Axel and DuAxel rovers for the sustainable exploration of extreme terrains. *Journal of Field Robotics*, 29(4):663–685, July 2012.

[5] Atsushi Kawakami, A Torii, Kazuhiro Motomura, and Shigeo Hirose. SMC Rover: Planetary Rover with transformable wheels. *Experimental Robotics VIII*, pages 157–162, 2003.

[6] Shimon Even. *Graph Algorithms*. Cambridge University Press, 2011.

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[8] Patrik Haslum and Héctor Geffner. Admissable Heuristics for Optimal Planning. *AIPS 2000 Proceedings*, pages 140–149, March 2000.

[9] P.E. Hart, N.J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[10] Mike Phillips, Maxim Likhachev, and Sven Koenig. PA*SE: Parallel A* for Slow Expansions. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

[11] Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-Heuristic A*. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2014.

[12] Venkatraman Narayanan, Sandip Aine, and Maxim Likhachev. Improved Multi-Heuristic A* for Searching with Uncalibrated Heuristics. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 78–86, 2015.

[13] Fahad Islam, Venkatraman Narayanan, and Maxim Likhachev. Dynamic Multi-Heuristic A*. In *IEEE International Conference on Robotics and Automation*, pages 2376–2382, Seattle, 2015.

[14] Juan P Gonzalez, Andrew Dornbush, and Maxim Likhachev. Using State Dominance for Path Planning in Dynamic Environments with Moving Obstacles. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, pages 4009–4015, 2012.

[15] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, number May, pages 3310–3317. IEEE Comput. Soc. Press, 1994.

[16] Anthony Stentz. The focussed D* algorithm for real-time replanning. *14th International Joint Conference on Artificial intelligence (IJCAI)*, (August):1652–1659, February 1995.

[17] Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, June 2005.

[18] Dave Ferguson and Anthony Stentz. Field D*: An Interpolation-Based Path Planner and Replanner. In *International Symposium on Robotics Research*, volume 28, pages 239–253, 2007.

[19] Hiroshi Noborio, Tomohide Naniwa, and Suguru Arimoto. A quadtree-based path-planning algorithm for a mobile robot. *Journal of Robotic Systems*, 7(4):555–574, August 1990.

[20] Hanan Samet. An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures. *NATO ASI Series*, F(40):51–68, 1988.

[21] Priyadarshi Bhattacharya and Marina L. Gavrilova. Roadmap-based path planning - Using the voronoi diagram for a clearance-based shortest path. *IEEE Robotics and Automation Magazine*, 15(2):58–66, June 2008.

[22] P Svestka and M. H Overmars. Probabilistic path planning. Technical report, Utrecht University, 1995.

[23] Steven M Lavalle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[24] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, May 2011.

[25] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2997–3004. IEEE, September 2014.

[26] Didier Guzzoni, Adam Cheyer, Luc Julia, and Kurt Konolige. Many Robots Make Short Work: Report of the SRI International Mobile Robot Team, March 1997.

[27] Y.U. Cao, A.S. Fukunaga, A.B. Kahng, and F. Meng. Cooperative mobile robotics: antecedents and directions. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 1, pages 226–234. IEEE Comput. Soc. Press.

[28] J. Peng. Coordinating Multiple Robots with Kinodynamic Constraints Along Specified Paths. *The International Journal of Robotics Research*, 24(4):295–310, April 2005.

[29] C. Domshlak and A. Nazarenko. The Complexity of Optimal Monotonic Planning: The Bad, The Good, and The Causal Graph. *Journal of Artificial Intelligence Research*, 48:783–812, 2013.

[30] Thierry Siméon, Stéphane Leroy, and JP Lauumond. Path coordination for multiple mobile robots: a resolution-complete algorithm. *Robotics and Automation, ...*, 18(1):42–49, 2002.

[31] David Silver. Cooperative pathfinding. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 117–122, 2005.

[32] Wenjie Wang and WB Goh. Multi-robot path planning with the spatio-temporal A* algorithm and its variants. *Advanced Agent Technology*, pages 313–329, 2012.

[33] P Švestka and MH Overmars. Coordinated path planning for multiple robots. *Robotics and autonomous systems*, 23:125–152, 1998.

[34] Sebastian Thrun, W Burgard, and Dieter Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. *Robotics and Automation, 2000. ...*, (April):321–328, 2000.

[35] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun. Collaborative multi-robot exploration. In *International Conference on Robotics and Automation*, number April, pages 476–481, 2000.

[36] Wolfram Burgard and Mark Moors. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[37] Jennifer Buehler and Maurice Pagnucco. A Framework for Task Planning in Heterogeneous Multi Robot Systems based on Robot Capabilities. In *AAAI Conference on Artificial Intelligence*, pages 2527–2533, 2013.

[38] RR Murphy, Michelle Ausmus, and Magda Bugajska. Marsupial-like Mobile Robot Societies. *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 1–14, 1999.

[39] Robin R Murphy. Marsupial and shape-shifting robots for urban search and rescue. *Intelligent Systems and their Applications, IEEE*, 15(2):14–19, 2000.

[40] K M Wurm, C Dornhege, P Eyerich, C Stachniss, B Nebel, and W Burgard. Coordinated exploration with marsupial teams of robots using temporal symbolic planning. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5014–5019, October 2010.

[41] Paul Tompkins. *Mission-directed path planning for planetary rover exploration.* Phd, Carnegie Mellon University, 2005.

[42] Christopher Cunningham, Heather L Jones, Jacqueline Kay, Kevin M Peterson, L William, and Red Whittaker. Time-Dependent Planning for Resource Prospecting. In *i-SAIRAS: International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2014.

[43] Nathan D. Otten, Heather L. Jones, David S. Wettergreen, and William L. Whittaker. Planning routes of continuous illumination and traversable slope using connected component analysis. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2015, pages 3953–3958. IEEE, May 2015.

[44] Peng Wu and Hehua Ju. Mission-Integrated Path Planning for Planetary Rover Exploration. *Journal of Software*, 8(10):2620–2627, October 2013.

[45] TW McLain and RW Beard. Trajectory planning for coordinated rendezvous of unmanned air vehicles. *Proceedings of the AIAA Guidance, Navigation and Control*, (August), 2000.

[46] Neil Mathew, Stephen L. Smith, and Steven L. Waslander. A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In *2013 IEEE International Conference on Robotics and Automation*, pages 3497–3502. IEEE, May 2013.

[47] Malika Meghjani and Gregory Dudek. Combining Multi-robot Exploration and Rendezvous. In *2011 Canadian Conference on Computer and Robot Vision*, pages 80–85. IEEE, May 2011.

[48] Malika Meghjani and Gregory Dudek. Multi-robot exploration and rendezvous on graphs. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5270–5276. IEEE, October 2012.

[49] C Judith Hilitch. Linear Skeletons From Square Cupboards. *Machine Intelligence 4*, pages 403–420, 1969.

[50] Pawel Zebrowski and Richard T Vaughan. Recharging Robot Teams : A Tanker Approach. pages 803–810, 2005.

[51] Pawel Zebrowski, Yaroslav Litus, and Richard Vaughan. Energy-Efficient Robot Rendezvous. In *Fourth Canadian Conference on Computer and Robot Vision (CRV '07)*, pages 139–148, 2007.

[52] Yaroslav Litus, Richard T Vaughan, and Pawel Zebrowski. The Frugal Feeding Problem : Energy-efficient, multi-robot, multi-place rendezvous. *IEEE International Conference on Robotics and Automation*, (April):27–32, 2007.

[53] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Distributed path consensus algorithm. Technical report, University of Pennsylvania, 2010.

[54] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Multi-agent path planning with multiple tasks and distance constraints. *2010 IEEE International Conference on Robotics and Automation*, pages 953–959, May 2010.

[55] R Ghabcheloo. Nonlinear coordinated path following control of multiple wheeled robots with communication constraints. *International Journal of Adaptive Control and Signal Processing*, 21(September):133–157, 2005.

[56] J.P. Desai, J.P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, 2001.

[57] Pramod Abichandani and Hande Y. Benson. Multi-vehicle path coordination under communication constraints. In *2008 American Control Conference*, pages 650–656, Seattle, June 2008. Ieee.

[58] Martijn N. Rooker and Andreas Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435–445, April 2007.

[59] P. Tournassoud. A strategy for obstacle avoidance and its application to multi-robot systems. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1224–1229. Institute of Electrical and Electronics Engineers, April 1986.

[60] GAS Pereira, AK Das, RV Kumar, and MFM Campos. Decentralized motion planning for multiple robots subject to sensing and communication constraints. Technical report, University of Pennsylvania, 2003.

[61] Alejandro R. Mosteo, Luis Montano, and Michail G. Lagoudakis. Multi-robot routing under limited communication range. *2008 IEEE International Conference on Robotics and Automation*, pages 1531–1536, May 2008.

[62] P. D. Spudis, K. R. Stockstill, W. J. Ockels, and M. Kruijff. Physical Environment of the Lunar South Pole from Clementine Data: Implications for Future Exploration of the Moon. 26:1339, February 1995.

[63] Paige D. A. Sefton-Nash E., Seigler M. A. Thermal Extremes in Permanently Shadowed Regions at the Lunar South Pole. Technical report, Jet Propulsion Laboratory, Los Angeles, 2013.

[64] W. C. Feldman. Fluxes of Fast and Epithermal Neutrons from Lunar Prospector: Evidence for Water Ice at the Lunar Poles. *Science*, 281(5382):1496–1500, September 1998.

[65] Peter Berkelman. Design of a Day/Night Lunar Rover. Technical report, Carnegie Mellon University Robotics Institute, Pittsburgh, 1992.

[66] Henrik Hargitai and Ákos Kereszturi. *Encyclopedia of Planetary Landforms*. Springer New York, New York, NY, 2014.