Network-Guided Multi-Robot Path Planning for Resource-Constrained Planetary Rovers

Ryan Luna, Alexis Oyama, Kostas Bekris

Computer Science and Engineering Department, University of Nevada, Reno - Reno, NV, USA
e-mail: {lunaryan,alexis.oyama}@gmail.com, bekris@cse.unr.edu

Abstract

Planetary exploration can benefit by the presence of multiple robots, which must be able to coordinate their paths and avoid collisions. This work proposes the use of a wireless network for the high-level path planning of multiple planetary rovers. In this setup, robots receive commands from the network nodes so as to maneuver between locations. Thus, robots can focus on tasks such as local obstacle avoidance and localization. This is desirable in space applications where robots are resource-constrained. Towards this objective, this paper presents a distributed path planning algorithm that is executed on the network nodes and provides collision-free paths for the robots on a precomputed roadmap. The method also aims to minimize the occurence of deadlocks and the time it takes for each robot to reach its goal. The approach follows a distributed constraint optimization formulation that lends itself to a decentralized, message-passing solution that is appropriate for network-guided navigation. Simulations are employed to evaluate the method's scalability and computational cost, as well as the quality of the resulting paths.

1 Introduction

Using multiple surface robots can be advantageous in the colonization of planetary bodies [1]. These advantages range from reducing the weight and size of each robot to increasing robustness by providing redundancy. Multiple lighter robots can potentially be transported to a planetary body over many launches, where each launch is less expensive due to the smaller cargo. A high level of redundancy is also important to ensure that a mission is at least partly successful. Multi-robot teams also allow for adaptability in mission objectives and even self-repair. They can be used for the mining of moons and asteroids, the construction of human habitats, the detection of valuable resources and astronaut support during manned missions. Because of these advantages, various efforts in multi-robot colonies have been considered in the past [2, 3] and ESA has included multi-robot teams in their vision [4].

When multiple robots, however, operate in the same environment, it becomes necessary to coordinate their paths. This paper proposes the use of a static wireless networks for the computation of the high-level path planning of the robots. In this setup, the robots receive motion commands from the nodes to follow and in this way focus on other tasks, such as local obstacle avoidance and localization. This is beneficial in space exploration where robots are multi-tasking, resource constrained and have a limited time to compute a path. The network can store a roadmap of high-quality paths between regular destinations of the robots; paths that will be safe or more efficient, and can be reused into the future. For example, one requirement might be that paths have to pass through regions that receive sunlight so as not to drain the batteries of the robots. Then the network can compute good quality paths along the roadmap because it can have access to the position of the robots through distributed communication protocols.

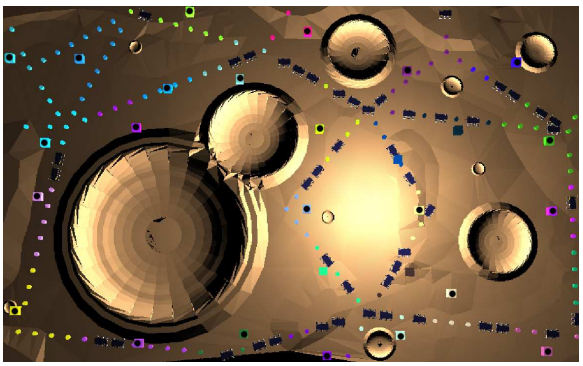

Figure 1. A snapshot of multiple simulated robots guided by a network. Each node has a different color and the part of the roadmap under its control is denoted with the same color.

A network could be already deployed in a planetary body for other uses, such as collecting seismological or meteorological data. There have already been works that study the feasibility and potential applications of wireless and sensor networks in space exploration [5, 6, 7, 8]. The challenge that this paper focuses on is the problem of using a network infrastructure to compute the paths of robots along a precomputed roadmap so as to avoid collisions, minimize the occurrence of deadlocks and the duration of paths.

1.1 Background

Multi-robot path planning can be approached with *coupled algorithms*, which plan for a single, composite system, and are complete. Nevertheless, they have exponential dependency on the number of robots. This led to coupled methods that reduce the size of the search space [9, 10, 11]. A coupled solution, however, is not realistic for network-guided navigation because it requires the collection of global information from all the network nodes. The focus here is to identify appropriate distributed algorithms, where each sensor has access only to local information (i.e., information within its 1-hop neighborhood) and does not flood the network with messages.

In *decoupled approaches*, path coordination can be achieved through a velocity tuning procedure that avoids collisions at path intersections [12, 13]. Alternatively, prioritized schemes compute paths sequentially for different robots in order of priority. High-priority paths are considered moving obstacles that must be avoided by lower priority ones [14]. While decoupled planners can solve problems orders of magnitude faster than coupled ones, they are inherently incomplete [15]. There have been techniques which aim to increase the reliability of decoupled planners by searching the space of prioritizations [16]. Alternatively, incremental planning [17] and schemes that employ coordination graphs [18, 19] have been used to ensure collision avoidance and deadlock reduction. For network-guided multi-robot path planning, it is necessary to consider sophisticated decoupled schemes, where coordination arises naturally from the constraints imposed on the robots, provides collision avoidance and minimizes the occurrence of deadlocks.

Network-guided navigation has been studied in the literature as a way to guide robots without a map or a need for robot localization [20]. The focus is often on the distributed computation of paths over a sensor network for an individual agent [21] by encoding dangerous regions as obstacles and often employing potential functions for navigation [22, 23]. Because flooding the network with messages is an issue, existing techniques try to minimize this effect by computing approximate shortest paths over a skeleton graph of the network [24].

This paper follows a *distributed constraint optimization* (DCO) formulation of network-guided navigation. One approach for DCO views it as optimal action selection in a multi-agent factored Markov Decision Processes [25]. This gives rise to algorithms for joint action selection based on linear programming [25] or asynchronous belief propagation [26]. Alternatively, ADOPT-based algorithms [27] provide a memory-bounded, asynchronous, localized way to search through task assignments with built-in termination. DCO problems can be also approached with auctions, which have been applied to multi-robot routing problems [28] and space applications [29].

1.2 Contribution

The approach in this paper is an online, distributed solution for network-guided multi-robot navigation without priorities, where the nodes have only local information. The nodes first compute multiple local paths for robots in their vicinity. Then neighboring nodes coordinate the assignment of paths to robots so that there is no collision and the robots progress towards their goals. This is achieved by formulating a DCO problem, where an optimization function is defined. This function depends upon unary payoffs, which represent individual path quality, and pairwise payoffs, which express interactions between robots' path. An anytime message-passing optimization protocol [26] is used to compute a good assignment given the available amount of time.

Simulated experiments confirm that the proposed approach avoids collisions for benchmarks where the coupled solution is computationally infeasible and the prioritized alternative quickly results in deadlocks. Passive or active deadlocks are rarely observed in the experiments for tens of robots operating in the same environment. Simulations are used to study the scalability, path quality and computational efficiency of the approach.

Most of the existing network-based navigation techniques [20, 21, 22, 23, 24] do not reason about interactions between multiple agents as this work does. Similarly, work on multi-robot path planning [9, 10, 11, 15, 16, 17, 18, 19] does not consider the constraints imposed by computing paths on a network. Moreover, the formulation in this paper falls between the extremes of coupled and decoupled approaches and only weakly constraints a robot's motion before considering interactions, which can be advantageous in providing a robust decoupled solution. At the same time, it lends itself to a message-passing protocol which is appropriate for a network-based solution. This work studies only an abstract version of the challenge, where the robots are moving on a roadmap, and does not incorporate various parameters, such as robot dynamics, bandwidth and throughput limitation and localization errors. This abstraction, however, allows to focus on the algorithmic and path planning aspects of the problem.

2 Problem Definition

Assume that static wireless nodes $\mathcal{N} = \{n^1, \ldots, n^{|\mathcal{N}|}\}$ are dispersed on the surface of a planetary body. The nodes can communicate among themselves as long as they are within a communication radius. The placement of nodes is random but they are able to form one connected network. A set of robots $\mathcal{A} = \{\alpha_1, \ldots, \alpha_{|\mathcal{A}|}\}$ is also present on the surface. The robots must navigate between various predefined destinations in the world. A roadmap of safe paths $G(V, E)$ for the robots is downloaded to the network nodes (an example is shown in Figure 1). This roadmap

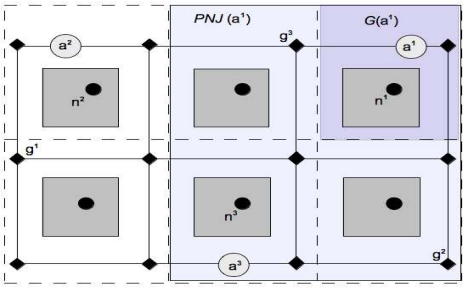

Figure 2. An illustration on a grid-like world. Node n^j has access only to its subgraph $G(n^j)$. The highlighted areas correspond to $\mathcal{LNG}n^j$.

connects all the destinations of interests. It has vertices that are collision-free positions and edges that correspond to collision-free paths. The vertex occupied by robot α_i at time t will be denoted as $v_i(t)$. Two robots cannot occupy simultaneously the same vertex or the same edge. Each robot has a goal vertex $g_i \in V$, which the robot aims to reach. The robots are able to communicate with at least one node from each vertex of the roadmap. Information regarding the roadmap is distributed over the sensor nodes. Each node n^j can detect the current location of robots (i.e., vertex on the roadmap) only if the robots are within radius.

In this setup, a robot does not compute its own path but receives commands from the nodes in order to reach its goal. A path π_i for robot α_i is a sequence of vertices: $\pi_i(0:t_m) = (v_i(t_0), \ldots, v_i(t_m))$. Given the path $\pi_i(0:t_m)$, denote as $e_i(t_j)$ the edge that robot α_i has to traverse in order to go from vertex $v_i(t_j)$ to vertex $v_i(t_{j+1})$. Two paths for two robots α_i and α_j are *compatible*: $\pi_i(0:t_m) \asymp \pi_j(0:t_m)$, as long as:

$$\forall\, 0 \le t \le t_m : v_i(t) \neq v_j(t) \wedge$$
$$\forall\, 0 \le t \le t_m - 1 : e_i(t) \neq e_j(t)$$

If one of the paths is of shorter duration than the other, then it can be expanded by repeating its last vertex. A *solution path* $\pi_i(0:t_m)$ for α_i is one where $v_i(t_m) = g_i$.

Then the problem of network-guided multi-robot path planning studied in this paper is the following:
Given the properties of sensor nodes $\mathcal{N}$ and robots $\mathcal{A}$ as described above, an initial placement of robots $\{v_1(0), \ldots v_{|\mathcal{A}|}\}$ and a set of goals $\{g_1, \ldots, g_{|\mathcal{A}|}\}$ on the roadmap $G(V,E)$, compute over the network and move the robots along paths $\{\pi_1(0:t_m), \ldots, \pi_{|\mathcal{A}|}(0:t_m)\}$ so that:

- *Each path is a solution path for the corresponding robot.*
- *Each pair of paths is compatible:*

$$\forall\, i, j \in [0, |\mathcal{A}|], i \neq j : \pi_i(0:t_m) \asymp \pi_j(0:t_m)$$

- *And t_m is minimized.*

3 Message-Passing Protocol

While there is at least one sensor with which an agent can communicate from every roadmap vertex or edge, many parts of the roadmap will be covered by multiple nodes. In order to identify the node that will be responsible for computing and communicating paths to each individual agent, the roadmap is partitioned into subgraphs $G(n^j)$ for each node n^j. During a preprocessing phase, each roadmap vertex is assigned either to the closest node or the one with the highest perceived signal strength. The union of all these subgraphs is the original roadmap, while their pairwise intersection is empty. This roadmap partition is available to the sensors before the online operation of the algorithm. It is similarly possible to partition the agents at each time step into subsets $\mathcal{A}(n^j)(t)$ based on the vertex they occupy: $\alpha_i \in \mathcal{A}(n^j)(t)$ iff $v_i(t) \in G(n^j)$.

3.1 Communication between Nodes and Agents

If a robot's goal is in a different subgraph than its initial location, then it must be guided by different nodes. Since nodes have often only local information, they cannot compute a complete path to the goal. Furthermore, as multiple robots enter and exit a node's local view, previously computed paths become invalid and they have to be recomputed. This suggests a replanning, partial solution, where nodes compute paths for robots at periodical intervals, preferably within the time it takes an robot to traverse an edge. One such interval will be referred to here as a cycle. During cycle $(t-1:t)$, nodes compute paths that the robots are going to execute during cycle $(t:t+1)$. Figure 3 explains the communication loop between robots and nodes during a cycle:

- A node n^j identifies robots $\mathcal{A}(n^j)$ in $G(n^j)$ (receives id, coordinates and goal coordinates).
- The nodes enter into a coordination procedure to compute compatible, solution paths for the robots. The coordination must be completed within the cycle's duration.
- Upon completion, each n^j transmits to $\mathcal{A}(n^j)(t)$ motion commands for the next cycle.

3.2 DCO Coordination Graph Formulation

Given the robots $\mathcal{A} = \{\alpha_1, \ldots, \alpha_{|\mathcal{A}|}\}$ at states $\{v_1, \ldots, v_{|\mathcal{A}|}\}$, the objective is to select an optimal joint assignment of paths $\{\pi_1, \ldots, \pi_{|\mathcal{A}|}\}$ that maximizes a global utility function Q decomposed into local utility functions:

$$Q(\mathcal{A}) = \sum_i Q_i(V(\mathcal{A}), \Pi(\mathcal{A}))$$

Q_i expresses the individual utility of α_i and potentially depends upon the α_i's interactions with all other robots. Often, however, a robot depends on a smaller subset of the team. An approach that exploits such dependencies involves a coordination graph $CG(V^C, E^C)$. In CG a node

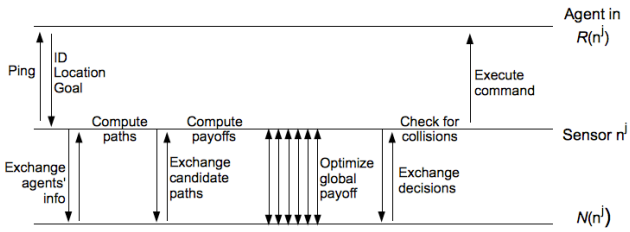

Figure 3. Communication between n^j, robots in $\mathcal{A}(n^j)$ and neighbors $\mathcal{N}(n^j)$.

represents a robot and an edge (i, j) represents that α_i and α_j are interacting. Then the global utility function can be decomposed as follows:

$$Q(\mathcal{A}) = \sum_{\forall i \in [0,|\mathcal{A}|]} f_i(\pi_i) + \sum_{(i,j)\in E^C} f_{ij}(\pi_i, \pi_j) \qquad (1)$$

where f_i is a unary payoff vector based on the paths α_i has available and f_{ij} is a pairwise payoff matrix that expresses the interactions between the paths of α_i and α_j.

To define the coordination graph at each cycle, it is necessary to respect the communication and information constraints imposed by the network. While node n^j has information only regarding its communication radius, there are also interactions between robots within n^j's local subgraph and outside it. But node n^j is constrained to exchange messages only with neighboring nodes on the communication graph, which are denoted as $\mathcal{N}(n^j)$. Thus, during each cycle, it considers information regarding robots only in its Local Neighborhood Graph $\mathcal{LNG}(n^j)$, the union of $G(n^j)$ and its neighbors' subgraphs:

$$\mathcal{LNG}(n^j) = G(n^j) \cup (\bigcup_{\forall n \in \mathcal{N}(n^j)} G(n))$$

The above requirements define the structure of the coordination graph: *"two robots share an edge in CG only if they are both located within an $\mathcal{LNG}(n^j)$ for a node n^j"*. This implies that neighboring nodes must exchange information regarding the robots within their subgraphs at each cycle. Overall, a node n^j implements the following procedure:

1. Exchange "ids, current locations, and goals" with $N(n^h)$ for all robots in $\mathcal{LNG}(n^j)$.
2. Generate a set of candidate paths $\pi_i, \forall\ \alpha_i \in \mathcal{A}(n^j)$.
3. Exchange π_i's with $N(n^h)$ for all robots in $\mathcal{LNG}(n^j)$.
4. Participate in the distributed optimization of Q
5. Exchange with $N(n^h)$ the paths p_i^* assigned by the optimization to robots within the $\mathcal{LNG}(n^j)$.
6. Check if the p_i^* lead to collision. If they do, enforce collision avoidance in the transmitted paths to $\mathcal{A}(n^j)$.

The following discussion provides details on the path generation step, how Q can be optimized and collision avoidance guarantees.

3.3 Generating Candidate Paths for the Robots

Since each sensor n^j has data only for the $\mathcal{LNG}(n^j)$, the paths for a robot $\alpha_i \in \mathcal{A}(n^j)$ must be also limited within the subgraph $\mathcal{LNG}(n^j)$. Thus, during each planning cycle, the possible destinations for α_i can be:

- all the leaves of the subgraph $\mathcal{LNG}(n^j)$
- and the robot's goal g_i, if g_i is contained in $\mathcal{LNG}(n^j)$.

The algorithm computes only the shortest path to each destination along the roadmap. Additionally, it is often necessary for robots to stop so as to let counterparts move in vertices ahead of them. This leads to the introduction of the "zero" path, according to which the robot does not move from its current position indefinitely. The "zero" path is undesirable but should be included so as to provide alternatives to other robots.

3.4 Message-Passing Protocol for Optimization

The *unary payoff function* $f_i(\pi_i)$ stores the utilities of paths for robot α_i as follows:

- For a path to goal g_i: $C - dt$, where C is an estimate of the maximum path length in the roadmap and dt is the path's length. The shorter the path, the higher the utility.
- Paths to a leaf vertex v_{leaf} of the $\mathcal{LNG}(n^j)$: $C - (dt + \|v_{leaf}, g_i\|)$, where $\|v_{leaf}, g_i\|$ is the length of the shortest path between v_{leaf} and g_i.
- The "zero" path gets a utility of 0.

A small noise value is added to all payoffs to avoid the creation of multiple local maxima in Q. Moreover, all paths that cause a robot to backtrack are penalized.

The *pairwise payoff function* $f_{ij}(\pi_i, \pi_j)$ expresses the pair-wise interactions between paths of different robots. For a pair $\pi_i(dt_i)$ and $\pi_j(dt_j)$ there are three outcomes:

- If compatible ($\pi_i(dt_i) \asymp \pi_j(dt_j)$, i.e., not colliding), the utility is the sum of the unary payoffs for π_i and π_j.
- If the paths collide during the next cycle, the utility is a highly negative constant.
- It the paths collide in t cycles ($t > 1$), then the sum of the unary payoffs is divided by a function inversely proportional to t $[max(e, e^{10-t})]$. Thus, imminent collisions reduce more the payoff than distant collisions.

Once the unary and payoff functions are computed, then all the necessary information is available to the nodes in order to optimize the global utility function. This can be achieved by a protocol that is analogous to the belief propagation algorithm in Bayesian networks, and which operates by iteratively sending messages $\mu_{ij}(\pi_j)$ for neighboring robots i and j in CG. Each sensor n^j produces for each robot $\alpha_i \in \mathcal{A}(n^j)$ and for all its neighbors $\alpha_j \in \mathcal{LNG}(n^j)$ the message $\mu_{ij}(\pi_j)$:

$$max_{\pi_i}\{f_i(\pi_i) + f_{ij}(\pi_i, \pi_j) + \sum_{\alpha_k \in \mathcal{LNG}(n^j)/\ j} \mu_{ki}(\pi(\alpha_i))\},$$

which is an approximation of the maximum payoff that α_i can achieve for path π_j of robot α_j, and is computed by

maximizing (over the paths of α_i) the sum of the payoff functions f_i and f_{ij} and all incoming messages to α_i except that from α_j. If α_i and α_j are both in $\mathcal{A}(n^j)$, then the sensor does not have to communicate the message to any other node and instead can internally update the message.

The sensors exchange messages until they converge. If CG happens to be a tree, then convergence to a fixed point is guaranteed in a finite number of steps. In graphs with cycles, there are no guarantees that the algorithm converges. In practice, however, it has been shown that the algorithm operates effectively even in graphs with cycles [26]. Here, it is sufficient if every sensor has an internal clock that marks the exhaustion of the available time for optimization at which point the sensor selects the best path at that iteration of the algorithm for each robot: $\pi_i^* = argmax(g_i(\pi_i))$, where $g_i(\pi_i) = f_i(\alpha_i) + \sum_{\alpha_k \in \mathcal{LNG}(n^j)} \mu_{ik}(\pi_i)$.

3.5 Guaranteeing Collision Avoidance

Since convergence is not guaranteed, it is possible that the final assignment $\{\pi_1^*, \ldots, \pi_{|\mathcal{A}|}^*\}$ contains incompatible paths. To detect an incompatibility, neighboring nodes exchange the resulting paths π_i^*. A robot involved in an incompatibility can be required to stop in order to avoid collisions. But this might result in a chain reaction, where a sequence of robots have to stop one after the other potentially across over the entire roadmap and flooding messages have to be exchanged.

If forcing a robot to stop is guaranteed to raise no additional conflicts, then this chaining effect can be avoided. To provide this guarantee, it has to be that during path generation the current position of every robot is considered as an obstacle by any other robot for the next cycle, i.e., if robot α_i occupies vertex v_i and robot α_j occupies the neighboring vertex v_j, then no path in the set π_j is allowed to have as its first vertex, vertex v_i. Then, each robot α_i can safely be stopped upon the detection of an incompatibility without its guiding sensor having to inform any neighbor.

The disadvantage of the solution is that it often causes robots to unnecessarily stop. The way to mitigate this effect is the following. A node n^j identifies robots α_i where their first step is to stop and the second step is to move to a vertex where another robot α_j is currently located. If robot α_j actually moves away from vertex v_j, then α_i does not have to stop and can accelerate the execution of its path by moving directly to v_j. What might happen is that multiple robots might be in the same situation as α_i, waiting for α_j to move out from vertex v_j. In this case a priority has to be used to decide which robot gets v_j. This improvement can also have a chaining effect if multiple robots are in consecutive vertices. The associated implementation in this paper ignores this chaining effects, as this operation is introduced only for efficiency and is not needed for collision avoidance.

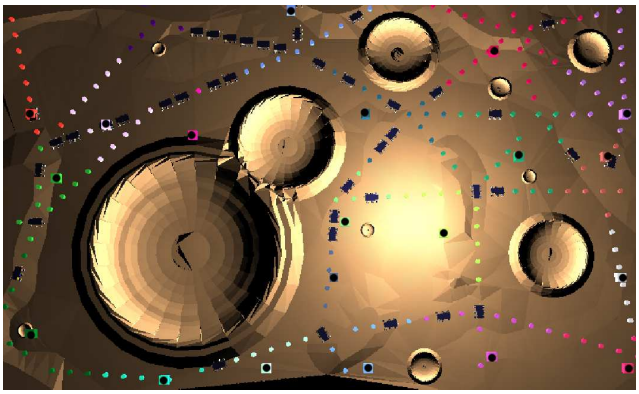

Figure 4. The second simulated roadmap. The simulation shows a 21 sensor network with 50 cooperating rovers.

4 Experiments

To show the feasibility of the proposed approach, a series of simulated experiments is conducted to evaluate different parameters, such as solution quality, computational efficiency, and scalability. The **setup** for these experiments involved the following parameters:

- Hardware: Parallel computing cluster made of Sun Fire X4100 M2 Nodes. Each node has two quad-core 2.6 GHz CPUs and 8GB of RAM.
- Software: The simulation is implemented using C++ for the sensor network and vehicle processes. All interprocess communication utilizes the Message Passing Interface standard (MPI), which guarantees lossless and in-order delivery of messages. Results are visualized in 3D, ensuring proper behavior of the sensors and rovers in the environment.
- Each node in the sensor network is simulated with its own process space and dedicated processing core in the parallel cluster. Because of the relatively low computational overhead for vehicles, all vehicles are simulated using a single process and core in the cluster.

Two different roadmaps on the same crater-like environment, shown in Figures 1 and 4, were utilized as **benchmarks** for the technique.

i) Sparser roadmap (Figure 1)
- 178 vertices and 183 undirected edges

ii) Denser roadmap (Figure 4)
- 228 vertices and 239 undirected edges

All **simulations** utilize a sensor network that provides coverage of all roadmap vertices. Each subgraph $G(n^j)$ is computed as the set of vertices that are closest in terms of Euclidean distance to the node n^j. For each of the roadmaps, a set of entry and exit vertices is defined for all rovers. In order to make the problem more challenging, constraints are placed on the selection of these points so that vehicle interactions are likely to occur during the simulation. The start and goal positions for each rover is

selected at random along the boundary of the roadmap. This ensures that rover interactions are almost impossible to avoid and require some level of coordination. When a rover enters the environment, the simulation must ensure that its start vertex is unoccupied at entry time. If the start vertex for a rover is occupied when it is to enter, a queue is formed for that vertex. As more vehicles attempt to enter the environment at the same position, they will also enter the queue. Vehicles are released from the queue as soon as the entry vertex becomes free.

The objective of the experiments was to evaluate the technique in terms of the following **metrics**:

1. *Path quality:* Evaluated through:
   - the steps each rover takes to reach its destination (compared against the length of the shortest path if interactions with other vehicles are ignored, which is a highly optimistic estimate),
   - the stops in an rover's path,
   - the times a rover is instructed to backtrack,
   - the times the optimization procedure results in a collision and collision avoidance has to be enforced,
   - the times the set of rovers reached a deadlock (all rovers stop moving).
2. *Computational Efficiency:* The duration of a planning cycle can impact the quality of the results and is an important parameter of the evaluation procedure.
3. *Scalability:* The number of sensors and vehicles in an experiment affect the algorithm's performance. The objective is to scale these numbers while minimizing the degradation in path quality or computational efficiency.

Intuitively, the number of stops in a rover's path is proportional to the number of other rovers operating in the environment and affects **path quality**. As the number of rovers increases, the likelihood that a rover will have to stop repeatedly when navigating to the goal also increases. Similarly, the number of backtracks performed by a rover is likely to increase, but this can be offset by penalizing reversing paths. In the more dense roadmap with a 28 sensor configuration and 25 robots, each robot stops for just a single time step during the entire path, and on average 1/3 of them perform a single backtrack. The same scenario with 100 vehicles sees a large jump in the average number of stops, to about 54 per vehicle. This indicates that the average rover is stopped for 54 time steps during navigation to allow other rovers to pass. Backtracks, however, do not increase by the same amount because of the backtracking penalty. Only 60% of rovers perform a single backtrack on average.

The jump in the number of stops has an effect on the total number of steps traveled when compared to the optimal single rover path. This ratio goes from about 1.4 in the 25 vehicle case to 3.2 in the 100 vehicle case. Deadlocks were not seen until the 75 rover scenarios in both roadmaps, and the deadlocks were in the smaller sensor network configurations. All three sensor configurations tested saw deadlocks with the 100 rover scenarios, with the 28 sensor configuration computing solutions in about 40% of experiments. Table 1 shows the percentage of deadlocks in varying experimental configurations. Finally, a high volume of rovers causes the network to intervene more in cases where the message passing protocol has not converged on a collision free set of controls. The 25 rover, 28 sensor case sees less than 1% of time steps needing extra intervention to avoid collisions between rovers. This percentage increases to about 20% in the 100 rover case.

Rovers	Sensors		
	14	21	28
25	0	0	0
50	0	0	0
75	100	2	0
100	100	100	58

Table 1. The percentage of deadlocks on the dense roadmap with varying numbers of rovers and sensors.

Because the technique is to be used in an online fashion, it is important that is **computationally efficient** and each planning step take a relatively small amount of time. This, however, can limit the quality of the paths between vehicles because the message passing protocol may not converge. The coordination portion of the approach is capped at 100 iterations of the message passing protocol, or 500ms, whichever comes first. These values work well in practice for situations involving up to 50 vehicles. As the number of rovers increases, it may become necessary to allow more time to the message passing protocol to allow a better solution to converge at runtime. In simulation, allowing more time for situations with many rovers decreases the deadlock rate and improves the solution quality. Table 2 shows a 75 vehicle scenario with two different message passing time values.

Different numbers of rovers and wireless sensors were tested using both the sparse and dense roadmaps to evalute **scalability**. Intuitively, scenarios involving a smaller number of rovers fared better than the more constrained simulations with more rovers. Simulations with a scattered sensor network with as few as 14 nodes successfully and consistently coordinated teams of up to 50 rovers through the environment. Increasing the number of sensors allows for more rovers to simultaneously operate in the same environment. With 21 sensor nodes, the actions of 75 rovers are easily coordinated in both roadmaps, and 28 nodes can find solutions for 100 rovers. Figure 5 shows the average total solution length computed for different numbers of sensors and rovers in the sparser roadmap. Similar values are computed for the more dense roadmap.

Time	21 Sensors 75 Rovers	Collision interventions	Path ratio
500ms	Sparse	21.16	2.40
	Dense	20.4	1.90
1500ms	Sparse	19.5	2.28
	Dense	18.2	1.88

Table 2. Experimental values showing the solution quality and the average number of collision intervention steps for different message passing times in both roadmaps.

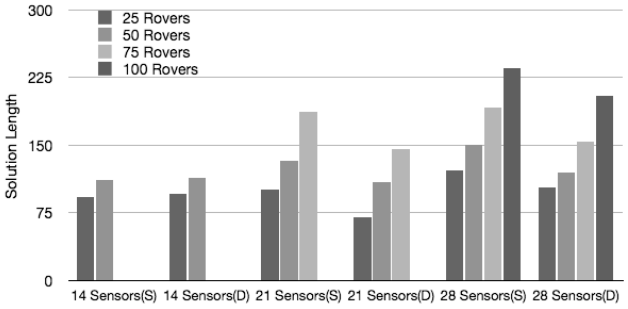

Figure 5. The average solution length (time steps) for varying numbers of vehicles and sensors in both roadmaps. Path coordination was limited to 500ms per solution step.

As the number of vehicles increases, the total solution length for each roadmap scales linearly. Experiments in both of the roadmaps stress the number of vehicles that the environment can support at any one time. For the 100 robot case, there were cases that 98 robots were moving simultaneously on the roadmap, which corresponds to a 50% occupancy of the underlying graph.

The number of sensors in the environment not only plays an important role in the number of rovers that can be successfully coordinated, but also the solution quality. The number and length of candidate paths generated for each rover is dependent on the sensor node's coverage area. In a uniformly distributed sensor network, too few sensors will result in large coverage areas and may introduce dependencies between rovers that are very far apart, resulting in less time for coordinating rovers that are closer together. For the dense roadmap and 14 nodes a high amount of deadlocks in scenarios involving 50 or more vehicles was observed because of the large amount of candidate paths generated for each rover. In the sparser roadmap, however, the 14 node network fared well for 50 robots. Too many sensors can also reduce the quality of paths. With many wireless sensors covering the environment, the coverage areas become very small, resulting in a small number of short candidate paths for each rover. Furthermore, dependencies between rovers are not introduced until they are very close, which can result in sub-optimal choices in path coordination or even deadlock situations. This can be seen by comparing the results for the same number of rovers with an increasing number of sensors.

5 Discussion

This paper proposes the use of a wireless network to guide multiple rovers that can potentially be used in space applications. An abstract version of the problem was studied, where robots move along a precomputed roadmap. This work proposed a distributed algorithm for this problem, where each node of the network has information about robots only in a local neighborhood and exchanges information only with 1-hop neighboring nodes. The algorithm casts the challenge as a distributed constraint optimization (DCO), models the interactions of agents through a coordination graph and applies a message passing protocol for the solution of the DCO problem. While it follows a decoupled approach for multi-robot path planning and given only local information on the network nodes, simulated experiments showed that deadlocks occur rarely on benchmarks where simple prioritized schemes failed very quickly.

There are many directions for further investigation of the idea of network-guided navigation for planetary rovers. (i) In particular, it is interesting to study how to construct and provide the roadmap to the nodes, which will be used in order to guide the robots. (ii) Moreover, instead of planning on a graph, it is possible to consider continuous space planning, or even taking dynamic motion constraints into account. Similarly communication constraints, such as bandwidth and throughput, or lcoation uncertainty can be included in the problem formulation. (iii) If all the agents are gathered in a small part of the workspace, then the corresponding nodes have a higher load. Load balancing schemes where heavy-loaded nodes outsource computation to less loaded neighbors can be helpful. Such load-balancing can also be achieved through the path planning process itself, where the nodes directing robots to different regions of the workspace so as to avoid congestion. (iv) Similarly, what should be the network's behavior if there are gaps in the coverage of the workspace or how should the network adapt when a node fails.

References

[1] J. Leitner, "Multi-robot cooperation in space: A survey," *Advanced Technologies for Enhanced Quality of Life*, pp. 144–151, 2009.

[2] P. S. Schenker, T. L. Huntsberger, P. Pirjanian, and E. T. Baum-Gartner, "Planeteray rover developments supporting mars exploration, sample return and future human-robotic colonization," *Autonomous Robots*, pp. 103–126, 2003.

[3] A. Schiele, J. Laycock, A. W. Ballard, M. Cosby, and E. Picardi, "Dalomis: A data transmission and localization system for swarms of microprobes," in *Intern. Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2005, p. 90.

[4] G. Visentin, "The ESA A&R technology R&D plan 2007-2009: Serving European Future Missions," in *9th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2008.

[5] R. Alena, J. Ossenfort, C. Lee, W. E., and T. Stone, "Design of hybrid mobile communication networks for planetary exploration," in *IEEE Aerospace Conference*, 2004.

[6] T. Sun, L.-J. Chen, H. C.-C., and M. Gerla, "Reliable sensor networks for planet exploration," in *IEEE Int. Conf. on Networking, Sensing and Control (ICNSC '05)*, Tucson, USA, March 2005.

[7] R. Newman and M. Hammoudeh, "Pennies from heaven: A retrospective on the use of wireless sensor networks for planetary exploration," in *NASA-ESA Conference on Adaptive Hardware and Systems*, 2008, pp. 263–270.

[8] E. Del Re, R. Pucci, and L. S. Ronga, "Performance Evaluation of an IEEE802.15.4 Wireless Sensor Network in Mars Exploration Scenario," in *Wireless Communication Society, Vehicular Technology, Information Theory and Aerospace/Electronics Systems Technology*, Aalborg, Denmark, 2009.

[9] P. Švestka and M. Overmars, "Coordinated path planning for multiple robots," *Robotics and Autonomous Systems*, vol. 23, no. 3, pp. 125–152, 1998.

[10] C. Clark, S. Rock, and J. C. Latombe, "Using dynamic robot networks for motion planning in multi-robot space systems," in *Proc. of the 7th Intern. Symposum on Artificial Intelligence, Robotics and Automation in Space*, Kobe, Japan, May 2003.

[11] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha, "Centralized path planning for multiple robots: Optimal decoupling into sequential plans," in *Robotics: Science and Systems V*, 2009.

[12] K. Kant and S. Zucker, "Towards efficient trajectory planning: The path-velocity decomposition," *International Journal of Robotics Research (IJRR)*, vol. 5, no. 3, pp. 72–89, 1986.

[13] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," *Int. Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, 2005.

[14] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," in *IEEE Intern. Conference on Robotics and Automation (ICRA)*, 1986, pp. 1419–1424.

[15] G. Sanchez and J.-C. Latombe, "Using a PRM planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2002, pp. 2112–2119.

[16] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning for teams of mobile robots," *Robotics and Autonomous Systems*, vol. 41, no. 2, pp. 89–99, 2002.

[17] M. Saha and P. Isto, "Multi-robot motion planning by incremental coordination," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 5960–5963.

[18] Y. Li, K. Gupta, and S. Payandeh, "Motion planning of multiple agents in virtual environments using coordination graphs," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2005, pp. 378–383.

[19] K. E. Bekris, K. I. Tsianos, and L. E. Kavraki, "A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 3784–3790.

[20] M. Batalin, G. S. Sukhatme, and M. Hattig, "Mobile robot navigation using a sensor network," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2004, pp. 636–642.

[21] Z. Yao and K. Gupta, "Distributed roadmaps for robot navigation in sensor networks," in *IEEE Intl. Conference on Robotics and Automation*, Anchorage, Alaska, USA, May 2010.

[22] Q. Li and D. Rus, "Navigation protocols in sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 1–33, 2005.

[23] P. Corke, R. Peterson, and R. D., "Localization and navigation assisted by cooperating networked sensors and robots," *International Journal of Robotics Research (IJRR)*, vol. 24, no. 9, pp. 771–786, 2005.

[24] C. Buragohain, D. Agrawal, and S. Suri, "Distributed navigation algorithms for sensor networks," in *IEEE Int. Conf. on Computer Communications (INFOCOM)*, April 2006, pp. 1–10.

[25] C. E. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored mdps," in *Proc. 14th Neural Information Processing Systems (NIPS-14)*, 2001, pp. 1523–1530.

[26] N. Vlassis, R. Elhorst, and J. Kok, "Anytime algorithms for multiagent decision making using coordination graphs," in *IEEE Transactions on systems, Man and Cybernetics*, The Hague, Netherlands, 2004.

[27] P. Modi, W. Shen, M. Tambe, and M. Yokoo, "ADOPT: Asynchronous Distributed Costraint Optimization with Quality Guarantees," *Artificial Intelligence*, vol. 161, no. 1-2, pp. 149–180, 2005.

[28] M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, S. Koenig, A. Kleywegt, C. Tovey, M. A., and S. Jain, "Auction-based multi-robot routing," in *Robotics: Science and Systems I*, 2005, pp. 343–350.

[29] M. B. Dias, D. Goldberg, and A. Stentz, "Market-based multirobot coordination for complex space applications," in *7th Intern. Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2003.