M.A.R.S.

Martian Autonomous Routing System

The Martian Autonomous Routing System (M.A.R.S.) is a suite of software written by Ross Blassingame, Shawn Polson, Robert Ballard, Josh Jenkins, Chandler Samuels, and Dean Moser at the University of Colorado Boulder as a part of their Senior Capstone Software Design Project. The Senior Capstone is a year-long course where students form teams of five or six, pair with a sponsor, and build software based on the sponsor's initial proposal. Our team paired with NASA's JPL, specifically Marcel Llopis, to build the pathfinding software. In the initial project proposal, it was required that the software be able to ingest:

- An elevation map of a section of the Martian terrain
- The rover start coordinates
- The rover end coordinates

It was also required that the software be able to find, if they exist:

1. A path from the start coordinates to the end coordinates, assuming that (1) the rover cannot climb hard climbs, (2) the rover cannot descend steep descents, and (3) that the rover cannot see too far ahead—the rover is essentially discovering the terrain as it moves.
2. A path from the start coordinates to the end coordinates, assuming that (1) the rover cannot climb hard climbs, (2) the rover cannot descend steep descents, and (3) that the rover is aware of all of the terrain—we can assume we have preloaded the entire elevation map into the rover's internal memory.

The first option is an exploration in which there could be backtracking involved, because the rover might try paths and end up in places where it cannot continue, whereas the second option involves the calculation of the ideal path upfront. In these initial project specifications, paths could be output as an ordered list of coordinates.

The central ability of M.A.R.S., then, is to discover paths in elevation maps. Elevation maps in our context are specifically GeoTIFFs. M.A.R.S accomplishes these tasks by leveraging the GeoTools toolkit, which is an open source Java library that provides tools for geospatial data, in conjunction with other Java code written from scratch by us. Using elevation data stored in the elevation maps, M.A.R.S. is able to find paths by employing pathfinding algorithms. Ultimately, the purpose of this is to aid in the design and planning of future rover missions by providing proposed paths, and to simulate a hypothetical future rover which is capable of autonomously maneuvering itself across the Martian surface. This handout's primary purpose, aside from describing M.A.R.S., is to compare the pathfinding algorithms employed in our software with the purpose of drawing conclusions about their respective advantages and disadvantages in guiding a rover.

The two metrics used to evaluate algorithms were length of the route and time taken to generate it. We were able to determine the following takeaways:

- A higher slope tolerance will result in a better (or equal) route and less calculation time.
- A heuristic is critical to quickly generating routes.
- A changing field of view generally decreases route length but makes generation take longer.

First, in terms of length, limited algorithms benefitted the most from an easier slope: their routes for the most challenging slope, 8º, were, on average, over 100km longer than the easiest slope, 32º. Unlimited algorithms benefitted from full information, and the 8º route was only 11% longer than the actual distance between the start and end points. In terms of speed, all but one of the limited algorithms ran faster, though their unlimited counterparts benefitted more: the 8º slope routes were, on average, a full 2633% slower to generate than 32º slope routes. For context, the average unlimited 8º route calculation took ~81 seconds per run, versus a 32º route taking only ~3 seconds. Also note that the Greedy algorithm was only used in Limited calculations, due to how it works. Unlimited Breadth-First Search and Dijkstra were also not included in these calculations for reasons explained in the next paragraph.

The problem faced by these algorithms is essentially one of pathfinding a particular sort of graph, wherein the complete elevation map is interpreted as a grid, with nodes at each pixel and edges to each neighboring pixel (when the slope is within a specified slope tolerance). This graph works out to having a massive number of nodes with, generally, an even greater number of edges connecting them. Because of this, a heuristic is essential to avoid getting bogged down by the size of the grid. This is represented through the unlimited versions of Breadth-First Search and Dijkstra: they do not use heuristics, and in comparison to the other algorithms, took an average of 3–4 hours to complete a route, as opposed to ~35 seconds.

Generally speaking, a greater field of view will reduce the length (or in other words, improve the quality) of a route, but will cause the algorithm to take longer to complete. On average, paths were 26% longer with the lowest field of view compared to the highest. In contrast, the highest field of view took just over twice as long as the lowest one. The exception to this is limited Breadth-First Search. It appears to have a "sweet spot" of around 5.54 km for its field of view, as all other fields of view tested consistently performed worse than that one, both in terms of time taken and route length.

From the data taken, we can conclude which algorithms performed the best and worst. There are four categories in which ranking can be made—time taken and route length, with respect to limited and unlimited fields of view. The algorithm that produced the best routes was Breadth-First Search, although A* produced similarly good results much more quickly. The fastest algorithms was Greedy. In the figure below are the exact results in these rankings. The rankings are based on median time taken with a precision of 1ms, and the best 8º slope route, averaging based on results from varying fields of view with a precision of 1 pixel, or 231 meters.

FOV Scope	**Data**	**1st**	**2nd**	**3rd**	**4th**	**5th**
Unlimited	**Length**	BFS (97.944km)	A* (99.561km)	Dijkstra (104.643km)	Best First (112.497km)	Greedy (209.517km)
Unlimited	**Time**	Greedy (0.005s)	Best First (0.016s)	A* (0.613s)	Dijkstra (hours)	BFS (hours)
Limited	**Length**	BFS (176.311km)	A* (202.645km)	Greedy (209.517km)	Dijkstra (221.414km)	Best First (236.295km)
Limited	**Time**	Greedy (0.003s)	Dijkstra (3.433s)	BFS (11.540s)	Best First (25.928s)	A* (32.081s)

This kind of software is going to become more and more prevalent as space science progresses and rovers become more autonomous. As humans venture further into space, autonomy will become imperative because of the infeasibility of manual control at long distances. This work helps pave the way for these kinds of breakthroughs in rover autonomy.