Exploring Mars

Rodrigo Ipince

April 30, 2008

Contents

1 Overview **2**

2 State Representation **2**

3 Action Representations **3**

4 Search Method **4**
4.1 Heuristic 4
4.1.1 Distance 4
4.1.2 Battery 5
4.1.3 Communication 5
4.1.4 Waiting 5

5 Performance **5**

6 Conclusion **6**

1 Overview

This paper tackles the problem of planning a rover's mission in Mars. The rover starts the mission at the lander with full battery. Its objective is to retrieve samples from a set of specified map locations, communicate with an orbiting satellite during specified communication windows, and return to the lander. Planning a mission involves determining the sequence of actions that the rover will follow to accomplish the mission.

The following are descriptions of the actions the rover can perform:

Drive The rover can move throughout a grid map (lattice plane) by driving to any of the 4 adjacent points to its current location. It can drive at any time of day. Driving incurs a time cost and a battery cost that may depend on the rover's current location.

Recharge The rover may choose to recharge its battery. Once it starts recharging, it must keep recharging until the battery is fully charged. The rover can only charge during the day. That is, if night falls during recharging, the rover will wait until daybreak and resume recharging. Each hour spent recharging results in a battery unit. A fully charged battery has 10 battery units.

Drill The rover can drill to take a scientific sample from its current location. This takes 1 hour and requires 1 unit of battery.

Communicate While on the mission, the rover must start communication in every communication window. These occur every day at 10 AM Mars time (see assumptions below) and last for 2 hours. Communicating takes 15 minutes and requires 1 unit of battery.

Wait The rover can sit and do nothing without consuming any battery.

None of these actions can overlap in time. Additionally, we assume the Mars day has 24 hours, daylight hours are from 6 AM to 6 PM, and the landing occurs at midnight, Mars time.

Given this setup, we tackled the problem from a search perspective. This paper describes the state and action representations used to model the problem, the search approach used, and an analysis of the time performance of the chosen method. In general, a minimalistic approach was used to model state and actions, in order to increase the performance of the search. The search method used was A* with a heuristic that incorporated four different estimates: time spent moving, recharging, communicating, and waiting.

2 State Representation

The state representation used is very simple. It consists of a list of assertions that can hold all the information about the state. The possible assertions are the following:

- `(action <action>)`: Indicates the action that led to this state.
- `(at (x y))`: Indicates the current plane location.
- `(time t)`: Indicates the time at which the rover reached this state.
- `(battery b)`: Indicates the current battery charge of the rover.
- `(need-sample (x y))`: Indicates that the rover needs a sample from location `(x y)`.
- `(have-sample (x y))`: Indicates that the rover holds a sample from location `(x y)`.
- `(need-communicate p)`: Indicates that the rover needs to communicate in period `p` (day).
- `(have-communicated p)`: Indicates that the rover has communicated in period `p`.

The goal state can then be represented as a list of `have` assertions, together with an `at` assertion to indicate the goal location, which should be the lander, located at (0,0). The goal state should not care about the rover's battery, the last action the rover took, nor the time at which the rover ended the mission.

Now, if the `action`, `time`, `battery`, and `need` assertions are not needed to encode the goal state, then why should we keep track of them in the state representation? It turns out that this information is needed for a variety of purposes within our search framework. The action is needed to reconstruct the search path; the time is needed to determine the outcome of certain actions (e.g., to recharge and communicate, we need to know whether it is daylight and whether we are within a communication window, respectively), to calculate heuristics, and to calculate the time-cost of the state; the battery is needed for the same reasons; the need assertions are needed to determine whether the robot cares about performing some actions (e.g., we don't want the rover to drill unless it needs to, because if it had the choice, the branching factor of the search tree would be huge and the search would take ages).

Lastly, notice that with this representation, the goal state must specify on what communication periods must the rover communicate, but this information may not be known a priori. To go around this issue, we just plan missions as if the rover did not need to communicate, and if the optimal plan ends up requiring communication to be in accordance with the problem definition, then we add the corresponding `need-communicate` assertions and replan.

3 Action Representations

The action representations (or procedures) were also designed with a minimalistic approach. We wanted to have the least amount of actions possible at any given state, so that the branching factor of the search tree would be as small as possible. Notice that it is not enough to have few action procedures. Each of these is a function of the current state and a set of arguments. The number of arguments and their domains also affect the branching factor, which is why we tried to make each action procedure take as few arguments as possible and make their domains as retrictive as possible.

Following this approach, we ended up having one action procedure per physical action, except for waiting. This is another key point. Since we are optimizing for time, there is no point in waiting unless the rover is forced to. This can happen if it needs to recharge during the night or if it is waiting for the next communication window. Thus, we eliminated the need of having an action procedure for waiting by incorporating the waiting effects into the recharge and communicate procedures.

With this in mind, we designed and implemented the following set of actions:

- `(go d)`: The go action enables the rover to drive from its current location (kept in the state) to a neighboring location. The direction argument `d` can assume the values `x-`, `x+`, `y-`, or `y+`, and specifies which ending location to go to. This the only action that takes an argument.

- `(drill)`: The drilling action enables the rover to take a sample from its current location. The rover must be in need of a sample from that location in order to be able to take one.

- `(recharge)`: The recharge action completely recharges the rover's battery. This action incorporates all the details involving the recharging specification. For instance, if recharge is performed during the night, then the rover will wait until daylight to recharge, so the resulting state could possibly be over 10 hours in the future.

- `(communicate)`: The communicate action makes the rover communicate only if it actually *needs* to communicate in the current period. If it tries to communicate while not being inside a communication window, then it will wait until the next available window and communicate then.

As a remark, notice that at each state, there is at most 7 things the rover could do: move in one of four possible directions, drill, recharge (and wait as needed in accordance with the problem definition), or communicate (and wait as needed).

There are some requirements that have not been accounted for with the current state and action representations. For instance, if a rover needs to communicate on period 3, but the communication window corresponding to that period has already passed, then it is impossible for the rover to complete the mission successfully. Instead of performing checks for this in each action and not allowing the rover to proceed once it has reached this state, we chose to keep things simple and account for this within our heuristic function. Thus, if the rover has failed to communicate, our heuristic estimate will be some insanely large number, which will make the search algorithm avoid such path.

4 Search Method

The search method used is A* with a consistent and admissible heuristic that is described in detail later. A* was chosen because of its speed in finding optimal paths and because the given problem allows the creation of good heuristics that can greatly speed the search if using a heuristic-based approach. Another approach could have been to use best-first search, but then the information about the actual cost of each state would have been lost. Since the heuristics we came up with were not too accurate in terms of estimating the actual cost to get to the goal, losing information about the actual cost would have made the search much more inefficient.

Since the heuristic used is both consistent and admissible, then it made no sense not to use an expanded list. Indeed, not using an expanded list resulted in endless (extremely long) searches, so there it was not too difficult to decide against not using one.

4.1 Heuristic

The heuristic used is made up of four components: distance, battery, communication, and waiting. Each of these components tries to estimate the time that the rover will spend performing the corresponding actions. The global heuristic just adds up all these estimates.

The components were designed in such a way that estimated remaining time was never accounted for more than once and such that all of them were underestimates of the actual cost. This ensures that the sum of the estimates will be below the actual cost, so the heuristic is admissible. Also, since each of these estimates depends on the current and goal nodes and the estimation function used is increasing as the current state approaches the goal, then the resulting heuristic is also consistent.

Detailed descriptions of each of the components are found in the following sections.

4.1.1 Distance

The distance component estimates the remaining time the rover will spend driving. It does so by first estimating the maximum distance it absolutely *needs* to cover. This is calculated in the following way: for each of the remaining drilling locations, the manhattan distance from the current location to the drilling location and from the drilling location to the goal location is calculated. If there are no remaining drilling locations, then only the manhattan distance from the current location to the goal location is considered. The estimate we are looking for is the maximum of these distances. Once this is calculated, we estimate the time cost of driving by multiplying the distance estimate with the minimum cost of driving.

It would be nice to use a better estimate than this one. A couple alternatives were considered. First, instead of considering the maximum distance from the rover's current location to a drilling site and from the drilling site to the goal, we could have used a path-based approach. We quickly decided against this since this is just a formulation of the travelling salesman problem. Solving that problem at every search node to calculate the heuristic value would be too wasteful and slow.

Second, we thought of using the actual cost of each path instead of the product of the minimum cost and the total distance. The problem here is that this is not well-defined; moving from location A to location B may be done through different paths that result in different costs. To choose which path to use, we could just take the shortest path, but again, this would involve solving another complicated problem that would take too long. Another option

was to choose any random path, but this means that the heuristic function would no longer be admissible, so this option was rejected too.

Now, notice that the estimate, in the way it is implemented, works best when the time cost of driving in the map is close to the minimum cost. If this is not the case, our estimate could be way below the actual cost and the heuristic is not that useful.

4.1.2 Battery

The battery component estimates the remaining time the rover will spend recharging. It does so by calculating the amount of battery units needed to complete the mission in terms of battery spent moving to the farthest drilling site and back to the goal, battery spent drilling, and battery spent communicating. This sets a bound on the minimum number of battery units necessary. If we subtract the current battery charge from this, then we get an estimate for the minimum number of extra battery units that the rover will need for the mission. Since it takes one hour to charge on unit of battery, this also corresponds to the minimum amount of time the rover will have to spend charging.

4.1.3 Communication

The communication component estimates the amount of time the rover will spend communicating. It does so by calculating the amount of communications are still needed to complete the mission and multiplying that by the communication time-cost.

As was mentioned in Section 3, if the rover has failed to communicate when it needed to, then an insanely large number (100,000 hours) is given as the estimate.

4.1.4 Waiting

Finally, the waiting component estimates the time that the rover will have to waste while waiting in the remainder of the mission. To keep things very simple, it only estimates a non-zero waiting time if the current time lies during the night. If so, then it calculates the ratio of the current amount of battery to the minimum amount of battery needed to do anything that is not waiting. This determines the maximum number of hours the rover can spend while not waiting. Thus, subtracting this number from the number of hours left until daybreak yields an estimate for the minimum amount of hours that the rover will spend waiting. Notice that we only consider the minimum amount of battery spend on a non-waiting action. Why? Waiting incurs a battery cost of 0, but if the rover decides to wait, then it will still waste time by waiting! Also, we make things simple by only considering times that are during the night. At these times, the rover cannot recharge nor communicate without waiting, so the minimum non-waiting battery cost is simply given by the minimum battery cost of driving.

This particular component of the heuristic could have been improved. For instance, if the rover has 4 units of battery at 4 PM and the minimum battery cost of driving, drilling, and communicating is 1, then we can conclude that it will have to waste at least 10 hours waiting. This is not incorporated into the waiting component and could potentially make a big difference. It was not incorporated since it is hard to determine the waiting outcomes at any arbitrary point of the day.

5 Performance

We performed tests in settings with different maps, costs, initial, and goal states. In this section, we only illustrate three of them. The following are descriptions of the three settings:

Test 1 Uniform map with minimum time and battery driving costs of 1 and 2, respectively; samples needed from (1 1) and (2 1); communication needed in period 0; default initial conditions (location (0 0), time 0, and battery 10).

Table 1: Performance (time, expanded) on three settings using different heuristics.

Setting	No Heuristic	Distance	Battery	Communication	Waiting	All
Test 1	637.9, 5,501	17.6, 644	17.4, 650	236.5, 3,233	663.1, 5,501	1.5, 107
Test 2	endless	endless	endless	endless	endless	326.5, 2,956
Test 3	endless	30.1, 1,073	570.8, 4,341	endless	endless	184.2, 2,677

Test 2 Same scenario as above, except that the minimum costs are now both 1, and there is an added drill location at (3 3).

Test 3 Map with minimum time and battery costs of 1, but where the rover is almost surrounded by high-cost terrain. It only needs to retrieve one sample, located at (3 0), which is also the goal location. However, the direct path to (3 0) is blocked by high-cost terrain. In spite of this, it is still better to go through the high-cost terrain to finish the mission (ticky map!).

For each setting, we ran the search procedure with no heuristic, using only one component of the heuristic, and using the complete heuristic. The results are shown in Table 1. The entries in each table correspond to the time taken, in seconds, followed by the number of nodes that were expanded. If an entry is "endless," it means that the search had not finished after over an hour.

From the table, we can see that the distance and battery components of the heuristic do a much better job than the other two. It is impressive how the search time dropped from 637 seconds to 1.5 in the first test when using no heuristic and when using the complete heuristic. It is also interesting to note that using only the waiting component yielded a worse result than when using no heuristic at all. The number of expanded nodes and the solution itself were the same. This means that the heuristic was not very useful and we had to pay the time cost of calculating it many thousand times.

However, notice that using the complete heuristic is not always the best option. The last test is a good example. In this case, the rover needs to overcome some rough terrain in order to get to its goal. When using the distance component only, the search is biased towards the goal location, which makes it end quickly. On the other hand, then using the battery component we get much worse results. This happens because since rough terrain incurs big battery costs. When the rover moves to a rough terrain location, it loses a lot of battery. However, its battery estimate is only decreased by a little, since we are only considering the minimum cost. Hence, it appears as if the rover had lost some units of battery in vain. Because of this, the search will try to avoid this path and go around the rough terrain. This makes the search take a lot longer, as can be seen in the table. The complete heuristic then gives a worse result than a distance-based heuristic by itself because the battery estimate plays a role in the complete one.

6 Conclusion

Using A* search with an admissible heuristic yielded some great results. It is really amazing how a heuristic can improve search time. But even though the heuristic improved the search time about 400-fold in some cases, we were still unable to get an answer for the setting that was provided by the staff. This might be because there was also some very high-cost terrain surrounding the drill locations in that map, which might have caused the battery component of the heuristic to steer us the wrong way.

That was another very interesting point. The conditions on the map greatly affect the best heuristic to choose. We saw how the battery component of our heuristic was detrimental on certain types of maps.

Finally, it would be interesting to further investigate the results of using a nonadmissible heuristic. A way in which this could be used is in calculating the estimated time spent driving, in order to better account for non-uniform maps.