Optimizing Map Exploration Parameters to Improve Coverage and Minimize Energy

Joseph Fair
jefair@ncsu.edu

Abstract

Map Exploration is a field that has uses in varied applications. From searching other planets to mapping our own ocean floor to modeling animal behavior, map exploration and finding efficient models has a wide range of applications. By implementing searches with only a few rules we can see intelligent behavior emerge. This project optimizes a tabu search algorithm for searching a map.

Keywords

Tabu Search - A search strategy that uses grid squares we've alredy visited to guide future decisions on direction. If we've already been to a grid square, we don't need to visit it again.
Chance Foreward - After deciding on a direction, this is the possibility of going perpendicular to the indicated direction. This allows us to avoid local minimums and find new terrain. Range is from 0 (very likely turn the wrong way) to 1 (never likely)
Direction Wiggle - After each direction decision, add a little variety to the direction we're headed. This keeps us from piloting the exact same path as last time, but allows for progress in the intended direction. Lower number should encourage going in the intended direction, but higher numbers should avoid getting death by boredom.
Grid Square - Maps are made up of grid squares. Each grid square has two attributes: Type and friction. Grid squares with more friction take more energy to cross.
Coverage - Percent of map that has been explored. As an objective, this is the percent that has NOT been explored in order to allow it to be minimized.
Energy - Crossing each grid square takes some energy, so we keep track of how much is expended.

Introduction

The concept of exploring an unknown area has been with us longer than computers, but automation gives us the opportunity to simulate conditions that are difficult and expensive in the real world. Knowing efficient algorithms for search helps us to find better value for time spent and better distance for the same amount of energy.

Related Work

- “Elephants avoid costly mountaineering“ J Wall, I Douglas-Hamilton, F Vollrath - Current Biology, 2006 - Cell Press
- "Navy to Explore New Ways to Employ Underwater Robots", Matthew Rusling, National Defense Magazine June 2009
- "New Tech Aims to Improve Communication Between Dogs and Humans" NC State News, https://news.ncsu.edu/2014/10/bozkurt-dogs-2014/

Model

Map - This is the 'world' in which the simulation lives. It contains a fully populated terrain and a set of agents that will explore the terrain.
Terrain - The terrain of the map is made of of grid squares of various types, and are set at creation time.
Agents - A map can have zero to many agents. For the purposes of this experiment, we are using one agent. However, using multiple agents is built in from the start. Agent also have a blank copy of the terrain so it can keep track of where it has been.

States: Excited, Bored, or Lazy
Agents can be in one of 3 states:
Excited - Agent is doing well and finding unknown grid squares. Continue tabu search.
Bored - Haven’t found any new grid squares for a while. Pick a random corner and go there.
Bored to Death - The final state (so to speak) of agents that exceed their boredom amount by double. Early termination.

Tick - Each agent is given a number of moves to cover the map, called ticks.
Friction - Each grid square has a type of terrain and a related energy cost for crossing that terrain.

Type	Friction
Paved	1
Grass	5
Slope	25
Cliff	100

Note: cliffs are never traversed, but they have to be taken into account when deciding the best path. That’s why they have the highest friction.

Problem Description

The problem here is twofold: minimize the number of unknown grid squares, and minimize the amount of energy required to explore the map.

Finding the Best Solutions

In order to find the best solutions, we compared two different optimizers: Differential Evolution and NSGAII

Differential Evolution (DE) is a process for generating solutions that are based on “generations” of solutions. We generate a random set of agents for the initial population and designate this “population zero”. We test each agent and measure the outcome for our objectives. Then, we generate a new population based on values of population zero. This is the crossover process, where attributes from teh parents effect what shildren are generated. Just to add a little chaos, we also allow the possibility of each solution changing a little at random. This is mutation. We compare each member of the new population with all the members of the old population and if the new one is better, we switch them out. This is selection. The agents left over are now called “generation one”. Then we do the same process (crossover, mutation, selection) until we either run out of comparisons or decide this solution is good enough.

Our mutations for this experiment are based on random selection from the generation (DE/rand/1), but other selections can include attributes only from the best solutions (DE/best/1) or mutations from more than one parent (DE/best/2). You can also choose based on which parents are closest, but this process is slower than picking the best or picking at random.

NSGAII is an algorithm that is popular today because it is effective and very fast. It starts out like DE in that it does crossover and mutation, but the selection process is different. First, it breaks solutions up into ‘bands’ of how well they did compared to other solutions. This is a fast way of weeding out the worst solutions. Drop all the furthest-out bands until your population is close to the original size. If there are still too many, you can take the furthest-out remaining band and do more time-consuming calculations on those.

This is a smarter way to go because you can do inexpensive evaluations like bdom to get an approximate solution, then more expensive solutions like cuboid space on the solutions that are on the line. In our experiments, NSGAII ran in about ??? compare final times ??? of Differential Evolution.

The Algorithm

Each agent follows the same algorithm to find a solution.
For each tick:

1. Decide on a best direction.
    a. If we're excited, do tabu search and determine a direction..
    b. If we're bored, find the 'bored direction'.
    c. If we're lazy, take the current direction and see if there is a low-energy alternative.
2. Move in that direction
    a. If there is a boundary or cliff, turn 90 degrees and try again (max 10 times).
    b. Otherwise, move to that spot
3. Look around and count the unknown spots you see from this spot.
4. Check your state.
    a. If you're excited and found at least one unknown, reset boredom to zero.
    b. If you're bored
        i. Check if you are near your corner. If so, be excited.
        ii. Check if you are bored to death. If so, terminate early.
        iii. Otherwise, increase your boredom count.

Problems Faced

One of the biggest problems was in trigonometry. I wrote part of the project assuming 0 degrees is East (to the right of the paper) and some assuming it was North to the top of the paper. This caused my later runs to go from 80% coverage to 20%. After some reflection, this was because for any particular direction, there's a good likelihood that instead of going that direction, the agent would turn in circles. The effect was most pronounced when it was in the bored state, because it would explore an area, get bored, and turn in circles until it died of boredom. The solution was to standardize on East as being direction zero, stop assuming any of my math worked, and write more tests for each function.

Data Results

First, we wanted to make sure that our DE and NSGAII were working properly. DE was first and we spent a good deal of time working with jMetal to set up the proper tests. However, no matter how we ran them, the tests only seem to be optimising in the x direction (percent coverage) and not the Y direction (energy used). Eventually we found the DE as implemented in jMetal is a single-objective problem. This was a surprise, but in hindsight I should have been able to tell because none of the results seem to be affected by the Y values of the initial population. Instead, I switched to GDE3, which is multi-objective, and my pareto plots normalized.

Effects of decisions

The range of percent results and energy used seemed to be a pretty good spread of values.
Distribution of percent Unknown values
NSGAII provided one map that had 11% unknowns. No maps were fully explored (0%)

```
rank ,         name ,    med   , iqr
--------------------------------------------------
   1 , Unknown_pct_Distribution_For_Q0 ,    21   ,    9 ( -- *-         |             ),11.58, 17.16, 21.05, 24.28, 28.41
   2 , Unknown_pct_Distribution_For_Q1 ,    39   ,    9 (       -- *-   |             ),31.81, 35.98, 39.78, 42.84, 47.06
   3 , Unknown_pct_Distribution_For_Q2 ,    57   ,    9 (             -|*-           ),50.72, 54.68, 57.48, 61.69, 64.98
   4 , Unknown_pct_Distribution_For_Q3 ,    78   ,   14 (              |    -  * ---  ),69.04, 74.15, 78.76, 84.65, 93.24
```

Distribution of energy used values

NSGAII shows a variance from 3 to 72. The low numbers probably resulted from an agent that got trapped early and died of bordeom after about 200 ticks.

```
--------------------------------------------------
   1 , Energy_Distribution_For_Q0 ,    8   ,    6 (- *-           |             ), 3.52,  6.32,  8.67, 10.70, 12.26
   2 , Energy_Distribution_For_Q1 ,    17   ,    4 (     -*-       |             ),14.26, 16.43, 17.92, 19.76, 21.59
   3 , Energy_Distribution_For_Q2 ,    29   ,    7 (         - *-- |             ),23.95, 26.86, 29.50, 33.04, 36.58
   4 , Energy_Distribution_For_Q3 ,    52   ,   21 (               |--  *   ----  ),40.94, 46.87, 52.62, 61.89, 72.41

rank          name      med      iqr
```

Decisions for Direction Wiggle
This shows that the lower half of the energy values are different from the upper half. That's probably because values in the upper half were more likely to go in the opposite direction

```
rank ,         name ,    med   , iqr
--------------------------------------------------
   1 , Energy_Vs_Direction_Wiggle_Distribution_For_Q0 ,    36   ,   33 (---   * -------|            ), 2.37, 20.22, 36.15, 51.67, 95.86
   1 , Energy_Vs_Direction_Wiggle_Distribution_For_Q1 ,    42   ,   69 (  ---  *      -|            ),15.22, 34.50, 42.89, 84.49, 99.39
   2 , Energy_Vs_Direction_Wiggle_Distribution_For_Q2 ,    84   ,   52 (    -----     *|-           ),26.11, 55.21, 84.49, 96.15, 101.07
   2 , Energy_Vs_Direction_Wiggle_Distribution_For_Q3 ,    96   ,   39 (       ------  |* --        ),47.22, 81.97, 96.15, 107.35, 123.19
```

This is a look at the inverse relationship. If the direction wiggle was low Q0, what's the average output of energy. This shows that direction wiggle did have a significant impact on how much energy was used. More wiggle means more energy wasted.

```
rank ,         name ,    med   , iqr
--------------------------------------------------
   1 , Direction_Wiggle_Vs_EnergyDistribution_For_Q0 ,    13   ,   12 ( -- * ----     |            ), 4.09,  9.23, 13.41, 18.64, 28.48
   2 , Direction_Wiggle_vs_Energy_Distribution_For_Q1 ,    18   ,   17 (  --  *  ------|            ), 6.83, 12.28, 18.38, 25.61, 42.89
   3 , Direction_Wiggle_vs_Energy_Distribution_For_Q2 ,    29   ,   21 (    ---    *  -|-----       ),11.89, 20.98, 29.59, 37.86, 54.67
   4 , Direction_Wiggle_vs_Energy_Distribution_For_Q3 ,    39   ,   32 (     -----     *    --------  ),14.84, 27.49, 39.72, 52.39, 71.93
```

Percent Unknown was also affected by direction wiggle, in that the higher Direction Wiggle, the less likely they are to get a percent complete.

```
rank ,         name ,    med   , iqr
--------------------------------------------------
   1 , Direction_Wiggle_vs_Percent_Unknown_Distribution_For_Q3 ,    33   ,   35 ( ----   *     -|---------     ),
   1 , Direction_Wiggle_vs_Percent_Unknown_Distribution_For_Q2 ,    34   ,   27 (   ---  *   ---|----         ),
   2 , Direction_Wiggle_vs_Percent_Unknown_Distribution_For_Q1 ,    57   ,   30 (      -----    |*  -----      ),
   3 , Direction_Wiggle_vs_Percent_Unknown_Distribution_For_Q0 ,    62   ,   27 (         ----- |  *  ----     ),
```

Decisions for Chance Forward
Similar to direction wiggle, the chance forward decision had an impact. Of the highest quartile of of chance forward, median energy distribution was only 17, but of the lowest, it was 39. It turns out that taking random turns away from the recommended path is NOT a good idea.

```
rank ,         name ,    med   ,  iqr
----------------------------------------------------
   1 , Chance_Fwd_vs_Energy_Distribution_For_Q3 ,    17   ,   17 ( ---  *  ----  |             )
   2 , Chance_Fwd_vs_Energy_Distribution_For_Q2 ,    19   ,   24 ( ---   *   ----|----         )
   2 , Chance_Fwd_vs_Energy_Distribution_For_Q1 ,    19   ,   20 (  ---  *   ----|--          )
   3 , Chance_Fwd_vs_Energy_Distribution_For_Q0 ,    39   ,   32 (    ------     *    --------  )
```

Decisions for Lazyness
The more lazy an agent is, the more likely he is to save energy. This table shows that the laziest agents have a small correlation with low energy

```
rank ,        name ,    med   ,  iqr
----------------------------------------------------
   1 , Laziness_vs_Energy_Distribution_For_Q3 ,    18   ,   20 ( ---  *   -----|---          ), 6.42, 11.59, 18.35, 27.92, 49.09
   2 , Laziness_vs_Energy_Distribution_For_Q2 ,    22   ,   26 (  ---   *     -|-------      ), 8.21, 16.18, 22.27, 37.09, 60.60
   2 , Laziness_vs_Energy_Distribution_For_Q1 ,    24   ,   26 (  ---    *    -|--------     ), 7.11, 15.18, 24.42, 37.84, 61.10
   2 , Laziness_vs_Energy_Distribution_For_Q0 ,    25   ,   25 (  ----   *   --|----         ), 7.95, 16.42, 25.30, 35.23, 52.62
```

Early termination vs normal termination
Early termination was the norm for the algorithm. Out of around 1050 evalutions, only 48 on DE and 55 on NSGII terminated after 10K ticks.

Of these, the median percent unknown ranged from 10 to 22, and the median energy ranged from 70 to 74. Otherwise the decisions matched the early termination numbers.

Other Measurments

In order to get an ideal pareto front, I ran the same test as before, but for 25 times the calculations. Comparing NSGAII to DE, I get IGD of 5.46, and if I reverse the comparison I get 4.40. The lower IGD indicates DE was marginally better. For spread, NSGAII was .70 and DE was .78, indicating NSGAII was slightly better.

For overall performance, I compared each ideal (25k calulations) to the ones I was analyzing above (1k calculations) and found

	NSGAII	DE
Population	50	50
Evaluations	1,000	20 generations

Initial IGD	258.3	129.2
Final IGD	5.3	1.84
I/F Ratio	51.6	70.21

Comparing the spread against 1K, 10K, and 25K evaluations, it doesn't look like the higher evaluations get very much better. I ran stats.py on it, but they all washed out with Knott-Schott of 1

Spread	NSGAII	DE
500 Evaluations/10 Generations	1.2	1.1
1000	0.9	1.0
10000	0.5	0.8

What does this tell us?

It looks like the biggest effect was the decision for Direction Wiggle, which makes sense because it applies to either bored or excited agents. Also, Chance Forward has an impact, but not as much as I'd thought. I originally assumed that Chance Forward would help avoid cul-de-sacs, but the bored state seems to have rendered that pointless. Laziness seems to have the least effect of the decisions. That makes sense, because it only applies when the agent is bored, and only some percent of the time.

Threats to validity

The terrain that was random, but uninteresting. Better terrain would have areas of sand, roads that were paved, and ranges of slopes that could be topped by other terrains. It would have forests, hills and valleys to impede visibility.

This algorithm was interesting, but there were many improvements we could have made to take it further:

Momentum: Right now any change in direction is immediate, but it would be interesting to see what happens if a change is not immediately taken, but affects a smaller increment. This would be similar to reducing the temperature in simulated annealing, where early runs are more likely to be random, and later ones more likely to focus in.

Path estimation: Instead of getting bored and running to a corner, you could run to a point on the nearest wall that is undiscovered. This would keep the agent from having to cross

most of the map (which may already be discovered) and get to a point where some good can be done.

Laziness: Even when in the excited state, an agent could try to conserve energy. This would allow the possibility of laziness to have a greater effect.

Conclusion

This report has taken a simple tabu algorithm and demonstrated some of the effects that come from different attribute settings. The most pronounced effects are in two areas: The decisions that are used the most and decisions that avoid making the agent go opposite the direction suggested. This is somewhat surprising, because most of the runs terminated early, which indicates the agent was bored to death. It seems like a bigger random factor would help avoid that problem.