CS 830: Program 1

Carmen St. Jean

February 11, 2013

1. *What is the size of the state space for this problem?*

   The size of the state space is $2^d \cdot n$ where d is the number of dirty cells and n is the number of unblocked cells.

2. *Explain the heuristic functions you devised and why they are admissible.*

   (a)

   $$h_1 = num_{dirts} + dist_{min}$$

   Where num_{dirts} is the number dirty cells remaining for the state and $dist_{min}$ is the Manhattan distance from the vacuum robot to the nearest dirty cell.

   This is admissible because the dirty cells will each have to be vacuumed, so there should be at least num_{dirts} number of actions for the robot. This number by itself is an extremely conservative lower bound because it does not take into account the actions required to move from cell to cell.

   Additionally, the robot will definitely have to visit the nearest dirty cell, which will take at least $dist_{min}$ actions. It is "at least" because there may be obstacles so there could be more actions required to reach the nearest dirty cell, so this is also a lower bound. This distance could be zero, meaning that the robot is on a dirty cell. If num_{dirts} is also zero, then the goal has been reached.

   (b)

   $$h_2 = num_{dirts} + dist_{min} + length_{MST}$$

   Where num_{dirts} is the number dirty cells remaining for the state, $dist_{min}$ is the Manhattan distance from the vacuum robot to the nearest dirty cell, and $length_{MST}$ is the length of the minimum spanning tree between the remaining dirty cells.

   The last heuristic, h_1, only considered the vacuum actions and the movement actions to the nearest dirty cell. This heuristic, h_2, takes into account the actions required to move from dirty cell to dirty cell while remaining admissible.

   In my implementation, every state has a minimum spanning tree for the remaining dirty cells associated with it, where the edges of the minimum spanning tree are the Manhattan distances between the dirty cells. The total length of this minimum spanning tree is a lower bounds on the number of movement actions required to visit all of the dirty cells. We can safely say it is a lower bound because minimum spanning trees are minimum cost subgraphs and there may be blocked cells that require the robot to take more actions to visit all cells.

   It is not possible for this heuristic to be overestimating the number of actions required. The robot will need to vacuum each dirty cell, it will need to "reach" that minimum spanning tree (potentially via the closest dirty cell), and it will need to visit every dirty cell in the minimum spanning tree.

   When there is only one dirty cell remaining, the length of the minimum spanning tree will be zero and the heuristic will be effectively the same as h_1. When no dirty cells remain, the length will also be zero and the entire $h2$ will equal zero.

   (c)

   $$h_3 = \begin{cases} num_{dirts} + dist_{min} + length_{MST}, & \text{if battery is sufficient} \\ num_{dirts} + dist'_{min} + length'_{MST} + num_{charges}, & \text{if battery is insufficient} \end{cases}$$

The h_3 heuristic is exactly the same as h_2 if the battery is sufficient. First, the h_2 heuristic is calculated for the state. $num_{charges}$ is calculated to be $\lfloor h_2/b \rfloor$ where b is the maximum battery charge. If $num_{charges}$ equals zero, then this means that the battery level will be enough to vacuum the dirty cells and h_2 is used.

However, if $num_{charges}$ is greater than zero, this means that the battery will need to be charged at least once to vacuum all of the remaining dirts. Therefore, the heuristic function recalculates a new minimum spanning tree for each charge cell, where the new trees each incorporate one charge cell. Whichever minimum spanning tree has the smallest value is saved as $length'_{MST}$. Then the heuristic determines the minimum Manhattan distance from the robot to either a dirty cell or the charging cell associated with $length'_{MST}$. This value is saved under $dist'_{min}$.

For the cases where the battery level is sufficient, then the heuristic is admissible because it is identical to h_2. For the other cases, it is also admissible because it takes into account at least one trip to a charge cell. By choosing the charge cell the makes for the smallest minimum spanning tree, we are ensuring that we have a lower bounds on how much distance will need to be travelled to reach the charge cell.

3. *Describe any implementation choices you made that you felt were important. Mention anything else that we should know when evaluating your program.*

   The states were represented as the robot's position in the vacuum world and a bit set indicating which of the dirty cells had been cleaned.

   For the implementation of my minimum spanning trees between the dirts, I hashed these based on the bit set indicating which dirty cells were represented in the tree and also the charging cell represented in the tree, if there was one. I did this to prevent regeneration of minimum spanning trees that were already created for other states with identical dirty cell configurations.

4. *What is the time and space complexity of each algorithm you implemented? Which algorithms are admissible?*

   Given a branching factor b, a maximum depth m, and a solution depth d:

Algorithm	Time Complexity	Space Complexity	Admissibility
depth-first	b^m	$b \cdot m$	No
depth-first-id	b^d	$b \cdot m$	Yes
ida-star	b^d	$b \cdot d$	Yes
uniform-cost	$b^d \cdot log(b^d)$ *	b^d	Yes
greedy	$b^m \cdot log(b^m)$ *	b^m	No
a-star	$b^d \cdot log(b^d)$ *	b^d	Yes

   * The log factor is to account for the insertion of each node into the open list, which is the Java PriorityQueue (which is implemented as a binary heap) and has log access time.

5. *Provide empirical results confirming your answers to the previous question.*

Results from small-1.vw

Algorithm	Time	Space	Nodes Generated	Nodes Expanded	Plan Length
depth-first	0.0 sec	980 KB	65	28	26
ida-star (with h1)	3.0 sec	980 KB	4484250	2111518	24
uniform-cost	0.0 sec	980 KB	1871	896	24
greedy (with h1)	0.0 sec	980 KB	2683	1301	24
a-star (with h2)	0.0 sec	980 KB	59	24	24

   In the above tests, depth-first-id failed to find a solution within one minute. As these results show, depth-first search is inadmissible, as it returned a longer plan length than the other search algorithms, which returned optimal plans.

Results from an 8x8 world with 25% cells blocked, 6 dirty cells, 2 charge cells

Algorithm	Time	Space	Nodes Generated	Nodes Expanded	Plan Length
a-star with h0	0.0 sec	980 KB	63423	21160	24
a-star with h1	0.0 sec	980 KB	12636	4212	24
a-star with h2	0.0 sec	980 KB	95	34	24
a-star with h3	1.0 sec	980 KB	95	34	24

As these results show, all my heuristics are admissible, with h_2 being significantly stronger than h_0 and h_1.

Results from an 10x10 world with 25% cells blocked, 20 dirty cells, 1 charge cell

Algorithm	Time	Space	Nodes Generated	Nodes Expanded	Plan Length
a-star with h2	3.0 sec	980 KB	896622	349596	73
a-star with h3	8.0 sec	980 KB	418561	160080	73

These results show that h_3 generates and expands less nodes than h_2 in more complicated worlds with batteries, but it takes more time because of the overhead of calculating the minimum spanning trees between the dirty cells and the charging cells.

Results from an 10x10 world with 25% cells blocked, 20 dirty cells

Algorithm	Time	Space	Nodes Generated	Nodes Expanded	Plan Length
a-star with h2	0.0 sec	980 KB	9511	4704	66
greedy with h2	0.0 sec	980 KB	207	104	78

These results show that greedy is much faster than a-star, but it fails to find an optimal solution, so it is inadmissible.

6. *What suggestions do you have for improving this assignment in the future?*

   We didn't learn most of the concepts we needed to know to complete this assignment until the second or so week, so we really had much less than three weeks to work on this assignment. Were we expected to read ahead on our own?