Lab Assignment 3
Single-Agent Search

T-622-ARTI,Artificial Intelligence, 2023-1

Reykjavik University - School of Computer Science, Menntavegi 1, IS-101 Reykjavík, Iceland

Jonas Hagg `jonash22@ru.is`
Philipp Zimmermann `philipp22@ru.is`
Jonathan Lutz `jonathan22@ru.is`

January 30, 2023

Abstract

This is the report for the third lab assignment of the artificial intelligence course.

1 Tasks

1.1 Task 1

A state is a named tuple consisting of the following information:

- Position of the agent
- Orientation of the agent
- A flag if the agent is turned on or off
- The locations of the dirty cells to clean

The location of the obstacles, the home location, the height, and the width of the environment are not part of the state because it is static information that does not change.

1.2 Task 2

With relatively few obstacles, the obstacle information is not considered in the estimate of the possible states. The estimate thus stays the same as in lab2:

The agent can be in $H \cdot W$ cells with every orientation $(H \cdot W \cdot 4)$. Besides, $\sum\limits_{n=0}^{D} \frac{!(HW)}{!(HW-n)}$ states exist for the dirty cells where D is the number of dirty cells at the beginning. This includes all possible combinations from a dirty room to a already cleaned room. Since the dirt can also be sucked, we compute the sum for up to D dirty spots. Also the agent can always be turned on or off (factor of two). Thus, the upper bound of the possible states is:

$$2 \cdot H \cdot W \cdot 4 \cdot (\sum_{n=0}^{D} \frac{!(HW)}{!(HW-n)}) \tag{1}$$

For a problem with fixed locations for the dirty cells, the D dirty cells can either contain dust or be cleaned already (2^D). Then, the upper bound of the possible states is:

$$2 \cdot H \cdot W \cdot 4 \cdot 2^D \tag{2}$$

1.3 Task 3

We are faced with a finite state space. Since we could have loops during state exploration, the search tree might be infinite if we do not detect re-visited states. We do not have constant costs per action.

1. BFS: Complete but not optimal (because step cost not always 1).
2. DFS: Neither complete (because potential infinite search tree) nor optimal. We can make DFS complete by detecting re-visited states (i.e. each state should be considered only once).
3. Uniform-Cost Search: Complete and optimal.
4. A*: Complete and optimal.

1.4 Task 4

In general, this depends on weather we detect re-visited states. If so, the size of the search tree must be smaller or equal to the size of the finite state space. If we do not detect re-visited states, however, the search tree can be infinite. In this case, the size of the search tree might be bigger than the size of the state space.

In this scenario, we assume the search tree to be smaller than the state space, because we can neglect certain states. For example, there is no need to consider a turned off vacuum if its location is not the home position. Moreover, for a concrete instantiation of the environment the dust position is fixed. Our space state estimate, however, assumes that the dust can be placed in any location. If the vacuum is placed is placed at the border of the grid, we can reduce the number of possible orientations because we do not want the vacuum to bump into the wall.

Overall, this analysis reduces the branching factor because it reduces the number of successors of a given states. A reduced number of successors reduces complexity of the algorithm.

1.5 Task 5

We assume that a goal state is defined by the fact that the agent has cleaned all dust, returned to its home location and is turned off. The heuristic is admissible because it computes the minimal number of actions required to reach this goal:

1. The dust cell located furthest away must be reached.
2. From this dust cell, the agent must return to its home location.
3. Perform d suck operations for d dirty cells left
4. Turn off the agent

The formula that has to be fulfilled for consistent heuristics can be changed to the following:

$$h(N) - h(N') \leq c(N, N') \tag{3}$$

A case distinction for all possible actions that can lead from N to N' with their associated costs $c(N, N')$ is given in the following:

1. *TURN_ OFF* The cost to turn off the robot in the home location is $1 + 50 \cdot D \geq 1$, not in the home location $100 + 50 \cdot D \geq 1$.
2. *TURN_ ON* The cost to turn on the robot is 1.
3. *SUCK* The cost to suck is 1 or 5 depending on whether the cell contains dust.
4. *GO* The cost to *GO* is always 1.
5. *TURN_LEFT* The cost to *TURN_LEFT* is always 1.
6. *TURN_RIGHT* The cost to *TURN_RIGHT* is always 1.

The cost of any action is greater or equal to 1. A case distinction for all possible $h(N)-h(N')$ is given in the following:

1. *TURN_ OFF* When turning off the robot, $h(N) - h(N') = 1$.
2. *TURN_ ON* The heuristic does not consider turning on the robot. Thus, $h(N)-h(N') = 0$.
3. *SUCK* If we suck dust, $h(N) - h(N') = 1$. If we suck on a cell not containing any dust, $h(N) - h(N') = 0$.
4. *GO* The *GO* action is only legal if not bumping into a wall. Depending on the direction, $h(N) - h(N') = 1$ or $h(N) - h(N') = -1$.
5. *TURN_LEFT* The heuristic does not consider turning left. Thus, $h(N) - h(N') = 0$.
6. *TURN_RIGHT* The heuristic does not consider turning right. Thus, $h(N) - h(N') = 0$.

As shown, $h(N) - h(N') \leq 1$ and $c(N, N') \geq 1$. Thus, the equation $h(N) - h(N') \leq c(N, N')$ always holds true. Therefore, the heuristic is consistent.

1.6 Task 6

For task 6, we implemented the A^* algorithm **without** the detection of revisited states. Our submitted implementation includes the A^* algorithm **with** detection of revisited states, the implementation of task 7. **Without** detection of revisited states, the search tree increases exponentially. With a branching factor of 4 (*TURN_LEFT*, *TURN_RIGHT*, *GO* and *SUCK*), and a solution depth of 42 for the small 5×5 *Grid 1*, the search tree size is 4^{42}. Due to limitations of the provided heuristic **SimpleHeuristic**, large parts of the search tree would have to be explored. Within ten minutes, the algorithm **without** detection of revisited states does not find a solution for any of the given grids.

We did several experiments with this basic algorithm and simplified the 5x5 grid by removing dust particles. If only one dust particle is placed on the 5x5 grid, the algorithm terminates within several seconds. The same holds for two dust elements. However, if we placed more than two dust particles on the grid, this limited version of the A^* algorithm did not terminate within several minutes. We observed this behaviour for different computers equipped with different CPUs. As elaborated above, we assume the reason for this lies in the size of the search tree which increases exponentially if we do not discard revisited states.

With the improved heuristic developed for task 9, the algorithm **without** detection of revisited states on *Grid 1* with three and four dust particles terminates in less than one minute.

1.7 Task 7

The detection of revisited states is implemented in the submitted code. As shown in 1.5, the heuristic is consistent and admissible. Thus, whenever a node is expanded we already found an optimal path to this node's state. As a result, we do not have to consider revisited states and do not suffer from an exponential search tree.

1.8 Task 8

For this task we used the A^* algorithm **with** detection of revisited states. We used the provided heuristic **SimpleHeuristic**. We observed the following results:

1.8.1 Obstacles_1 (Grid 1)

1. the number of state expansions
   110065

2. the maximum size of the frontier (measured in number of nodes)
   22948 nodes

3. the cost of the found solution (path cost)
   42

4. the run-time of the search (in seconds)
   1.5s

1.8.2 Obstacles_2 (Grid 2)

1. the number of state expansions
   122670

2. the maximum size of the frontier (measured in number of nodes)
   20496 nodes

3. the cost of the found solution (path cost)
   46
4. the run-time of the search (in seconds)
   1.94s

1.8.3 Obstacles_3 (Grid 3)

1. the number of state expansions
   1827644
2. the maximum size of the frontier (measured in number of nodes)
   25240 nodes
3. the cost of the found solution (path cost)
   286
4. the run-time of the search (in seconds)
   88.75s

1.8.4 Obstacles_4 (Grid 4)

1. the number of state expansions
   962334
2. the maximum size of the frontier (measured in number of nodes)
   37276 nodes
3. the cost of the found solution (path cost)
   202
4. the run-time of the search (in seconds)
   19.38s

1.8.5 Obstacles_5 (Grid 5)

1. the number of state expansions
   1258885
2. the maximum size of the frontier (measured in number of nodes)
   42507 nodes
3. the cost of the found solution (path cost)
   170
4. the run-time of the search (in seconds)
   27.17s

1.8.6 Obstacles_6 (Grid 6) - Goal State can not be reached!

The A^* algorithm did not find a solution to this problem, because one dust cell in the bottom right corner is not reachable. It terminates with the exception ”no plan found”.

1. the number of state expansions
   1046528

2. the maximum size of the frontier (measured in number of nodes)
   21923 nodes
3. the cost of the found solution (path cost)
   -
4. the run-time of the search (in seconds)
   24.47s

1.8.7 Obstacles_7 (Grid 7)

1. the number of state expansions
   106586
2. the maximum size of the frontier (measured in number of nodes)
   2029 nodes
3. the cost of the found solution (path cost)
   222
4. the run-time of the search (in seconds)
   1.35s

1.8.8 Obstacles_8 (Grid 8)

1. the number of state expansions
   223369
2. the maximum size of the frontier (measured in number of nodes)
   3336 nodes
3. the cost of the found solution (path cost)
   259
4. the run-time of the search (in seconds)
   3.09s

1.8.9 Obstacles_9 (Grid 9)

1. the number of state expansions
   451568
2. the maximum size of the frontier (measured in number of nodes)
   6203 nodes
3. the cost of the found solution (path cost)
   270
4. the run-time of the search (in seconds)
   7.02s

1.8.10 Obstacles_10 (Grid 10)

1. the number of state expansions
   886535
2. the maximum size of the frontier (measured in number of nodes)
   12729 nodes
3. the cost of the found solution (path cost)
   271
4. the run-time of the search (in seconds)
   16.27s

1.9 Task 9

The new Heuristic is calculated as follows: The Manhattan Distance from the current location of the agent to the closest Dirt Cell is used. For all remaining dust particles, the Manhattan Distance from one dust particle to the next closest dust particle are accumulated. The Manhattan Distance from the last Dirt Cell to the home location of the agent is added as well. This calculates the total cost of a route from the current location to home while collecting all Dust in an optimal way. It is still an underestimate of the real cost, because we do not consider the obstacles in the way. Like the **SimpleHeuristic**, it also adds one for each dirt left to perform the suck action and one more to turn the agent off, if it is turned on.

The cost of the found solution is identical to the implementation of 1.8, because we take the optimal path in both solutions. The number of state expansions decreased heavily (by 98% for Obstacle_1). This is because our heuristic is more accurate (dominant) than the SimpleHeuristic, therefore many more states can be excluded while building the tree. The maximum size of the frontier decreased as well as the run-time, because of the factors above.

1.9.1 Obstacles_1 (Grid 1)

1. the number of state expansions
   1369
2. the maximum size of the frontier (measured in number of nodes)
   185 nodes
3. the cost of the found solution (path cost)
   42
4. the run-time of the search (in seconds)
   0.05s

1.9.2 Obstacles_2 (Grid 2)

1. the number of state expansions
   1658
2. the maximum size of the frontier (measured in number of nodes)
   134 nodes
3. the cost of the found solution (path cost)
   46

4. the run-time of the search (in seconds)
   0.08s

1.9.3 Obstacles_3 (Grid 3)

1. the number of state expansions
   1155530
2. the maximum size of the frontier (measured in number of nodes)
   26022 nodes
3. the cost of the found solution (path cost)
   286
4. the run-time of the search (in seconds)
   54.75s

1.9.4 Obstacles_4 (Grid 4)

1. the number of state expansions
   121727
2. the maximum size of the frontier (measured in number of nodes)
   6316 nodes
3. the cost of the found solution (path cost)
   202
4. the run-time of the search (in seconds)
   5.72s

1.9.5 Obstacles_5 (Grid 5)

1. the number of state expansions
   272048
2. the maximum size of the frontier (measured in number of nodes)
   19172 nodes
3. the cost of the found solution (path cost)
   170
4. the run-time of the search (in seconds)
   12.47s

1.9.6 Obstacles_6 (Grid 6) - Goal State can not be reached!

The A^* algorithm did not find a solution to this problem, because one dust cell in the bottom right corner is not reachable. It terminates with the exception ”no plan found”.

1. the number of state expansions
   1061342
2. the maximum size of the frontier (measured in number of nodes)
   14984 nodes

3. the cost of the found solution (path cost)
   -

4. the run-time of the search (in seconds)
   47.37s

1.9.7 Obstacles_7 (Grid 7)

1. the number of state expansions
   79739

2. the maximum size of the frontier (measured in number of nodes)
   1995 nodes

3. the cost of the found solution (path cost)
   222

4. the run-time of the search (in seconds)
   2.19s

1.9.8 Obstacles_8 (Grid 8)

1. the number of state expansions
   177037

2. the maximum size of the frontier (measured in number of nodes)
   3326 nodes

3. the cost of the found solution (path cost)
   259

4. the run-time of the search (in seconds)
   5.48s

1.9.9 Obstacles_9 (Grid 9)

1. the number of state expansions
   344589

2. the maximum size of the frontier (measured in number of nodes)
   6538 nodes

3. the cost of the found solution (path cost)
   270

4. the run-time of the search (in seconds)
   12.28s

1.9.10 Obstacles_10 (Grid 10)

1. the number of state expansions
   605402

2. the maximum size of the frontier (measured in number of nodes)
   12293 nodes

3. the cost of the found solution (path cost)
   271
4. the run-time of the search (in seconds)
   24.23s