CS5011: Assignment 2 - Search - Rescue Simulations

Assignment: A2 - Assignment 2

Deadline: 2 November 2017

Weighting: 25% of module mark

Please note that MMS is the definitive source for deadline and credit details. You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

1 Objective

This practical aims to implement and evaluate a number of AI search algorithms applied to the task of search and rescue in the event of a disaster.

2 Competencies

- Design, implement, and document AI search algorithms.
- Compare the performance of AI search algorithms.
- Test methods for optimisation of AI search.

3 Practical Requirements

3.1 Introduction

When a country is hit by a natural disaster, such as an earthquake, timely response is critical. The first step involved in disaster response is rescue and evacuation. Among other tasks this involves evacuation of buildings where routes may be blocked by various obstacles and collapses. Autonomous robots have the potential to intervene in these situations by searching disaster areas for victims, to avoid further danger to human rescue teams. Testbeds and simulators have been developed in recent years for robots and intelligent agents attempting to solve this problems. For example, the RoboCup Rescue League[1] is a well known robot competitions for rescue operations.

[1] `http://www.robocup.org/domains/2`

3.2 The task

For this practical, we consider the problem of a single agent simulating a robot that navigates a building badly affected by an earthquake. We refer to this agent simply as a robot. The robot is equipped with a map of one building floor to be searched as shown in Figure 1. The robot aims to locate a victim and transport him/her to a safe location on the map. We call the victim Bob. The robot needs to move through the floor in the most efficient way to increase the chance of survival of the victim. Which algorithm should the robot use?

The floor map is laid out as a 10x10 grid as in Figure 1 and the robot moves one cell at a time, and to the adjacent cells only in the North, South, East, or West directions.

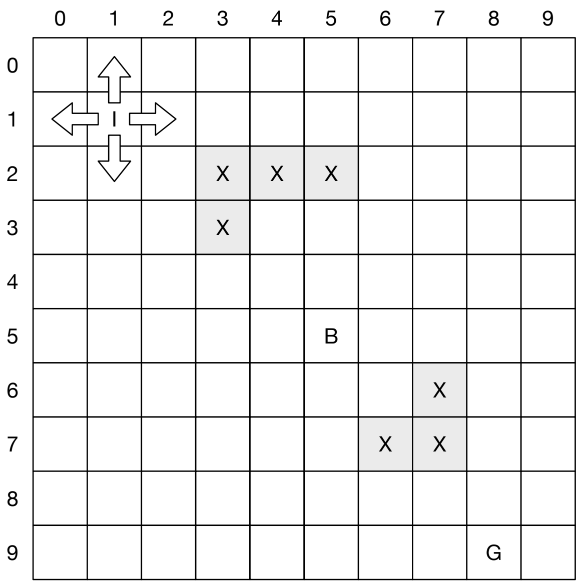

Figure 1: Example map for part 1 and 2

The robot starts from a cell marked as I, reaches the victim Bob in a cell marked B, and takes him to a safe goal position in a cell marked G. However, some of the cells are blocked by obstacles marked by X. The task is to find a route from I to B, then from B to G avoiding obstacles.

The file *maps.txt* contains six grids with different positions of I,B,F and obstacles. The grids are represented as Java arrays filled with chars, where each char represents a cell with the following code:

- 'O' means the cell is clear,
- 'X' represents an obstacle,
- 'I' represents the robot's initial position,
- 'B' is Bob's (the victim) position,
- 'G' is the safe goal position.

The aim of the practical is to implement and evaluate a set of AI search algorithms. There are two main criteria for evaluating search algorithms: the quality of the solution (e.g., the cost/length of the robot route), and efficiency represented here by the number of search states visited by the algorithm.

3.3 Part 1

Implement depth-first search (DFS), and breadth-first search (BFS). For each algorithm, once it reaches its goal it must be able to construct the path it has found from the starting point to Bob, to the goal. Your implementation should follow the general algorithm for search considering the order that new states are added to the list. The algorithm should also avoid loops and redundant paths. Please ensure that your implementation makes minimal changes between DFS and BFS. Your implementation should print out enough data to demonstrate that it works correctly. Make sure you print the current node, the list of states expanded, and the frontier at each step in the search, as well as the current solution details.

Points to cover in your report:

1. Describe the state space, initial state and goal, successor functions, and actions in this search problem, path cost, and details of their respective implementation.
2. Describe the implementation of the search strategy clearly, and the differences between DFS and BFS in your implementation.
3. Describe your choice(s) of order to add new states to the list.
4. Describe how you tested your implementation.
5. Evaluate BFS and DFS for this problem by running them on the data provided. Which algorithm is best in terms of the number of search states visited? Which algorithm produces the best (shortest) routes on the basis of path cost?
6. Include example runs of both DFS and BFS on some given maps, and use these for comparing the two algorithms. You could include your own example maps, after testing with those given.

3.4 Part 2

For many search problems, better results can be obtained by using a heuristic to choose the state to explore next. Implement best-first search (BestFS) and A* search, using two individual different heuristics and a combination of them with both BestFS and A*. As in Part 1, your implementation should follow the general algorithm for search.

Points to cover in your report:

1. Describe the heuristics chosen and the respective implementation.
2. Describe the search algorithms in your implementation.
3. Describe how you tested your implementation.

4. Evaluate BestFS and A* for this problem by running them on the data provided. How do they compare to DFS and BFS in terms of number of states visited and path cost? Which of the algorithms and which of your heuristics is best in terms of the number of search states visited? Which algorithm produces the best (shortest) routes on the basis of path cost? You could include your own example maps, after testing with those given.

3.5 Part 3

You may consider one of the following extensions:

1. Find out about bidirectional search, implement this approach and evaluate it in comparison with the results of part 2.

2. Consider that at the same position of Bob, there is a dog and a cat. The three of them are initially on the cell marked B, and can only be picked up at B and dropped off at G. All of those should be rescued by the robot, but the robot can only transport **one** of them at a time to the safe goal cell G. The problem is that Bob cannot stay alone with the dog as he is afraid of dogs, and the dog cannot stay alone with the cat as they don't get along with each other. This means that the following states are illegal:

   - only Bob and Dog in Cell B
   - only Dog and Cat in Cell B
   - only Bob and Dog in Cell G
   - only Dog and Cat in Cell G

   How can the robot bring everyone to safety? In addition to the path search, define the search components for this problem. Use one or more search algorithms and a map of your choice among those studied in part 1 and 2 to find a solution. The final solution should print the full journey and show the robot and show how to Hint: you may have to combine two searches in this task.

3. Consider that a team of 4 robots (R1-R2-R3-R4) is employed to rescue 4 victims (Ba, Bb, Bc, Bd) starting from positions I1-I2-I3-I4 and bringing them to the safe positions Gx-Gy-Gw-Gv as shown in Figure 3. At each step, all robots must move of one cell, but each robot is free to move in a direction different from those of the other robots. The problem is that the robots have to stay together. This means that the robots are in a configuration where each robot is in a cell adjacent to at least 1 cell that contains a robot. Furthermore, consider that among those legal, the least costly configuration is a square configuration where the robots occupy 4 cells such that each robot is in a cell adjacent to exactly 2 cells that contain robots. Figure 2 shows some examples of legal configurations. Define the search components for this problem, making sure that the move of each individual robot is indicated. Use an informed search algorithm of your choice among those studied in part 2 to find a solution for the map given in Figure 3.

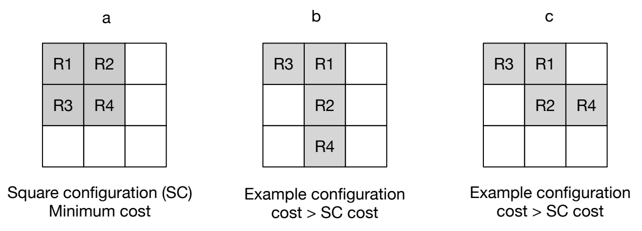

Figure 2: Example legal team configurations for Part 3.3

	0	1	2	3	4	5	6	7	8	9
0	I1	I2								
1	I3	I4								
2				X	X	X				
3				X						
4						Ba	Bb			
5	X	X				Bc	Bd			X
6								X		
7							X	X		
8									Gx	Gv
9					X				Gy	Gw

Figure 3: Map for Part 3.3

4 Code Submission and Running

The program must be written in Java and your implementation must compile and run on the School lab Machines. Please do not use libraries that implement search algorithms. A jar file is to be submitted for each part of the three parts attempted.
Name the jar file for Part 1 as "Search1.jar" and ensure it runs using:

```
java -jar Search1.jar [any param]
```

Name the jar file for Part 2 as "Search2.jar" and ensure it runs using:

```
java -jar Search2.jar [any param]
```

Name the jar file for Part 3 as "Search3.jar" and ensure it runs using:

```
java -jar Search3.jar [any param]
```

5 Report

You are required to submit a report describing your submission, with a limit of 4500 words excluding references. The report should include:

1. A list of the parts implemented and any extension attempted
2. If any of the functionalities is only partially working, ensure that this is discussed in your report
3. Literature review: a short literature survey on search algorithms and applications
4. Design: A description and justification for the mechanisms you have implemented as part of your solution. Please discuss the items listed for each part of the practical that was attempted.
5. Examples and Testing: Examples of the main functionalities and your approach to testing the system.
6. Evaluation: A critical analysis of the functionalities of your system and what can be improved.
7. Running: Include clear instructions on how to run your system
8. Bibliography: List all the references you cite in your literature review and elsewhere in your report and code.

More details on points to be discussed under sections 4–6 are listed in the description of each part of this assignment. Please include a word count in your report.

6 Deliverables

A single ZIP file must be submitted electronically via MMS by the deadline. Submissions in any other format will be rejected.

Your ZIP file should contain:

1. A PDF report as discussed in Section 5
2. One, two or three jars depending on the tasks achieved as discussed in Section 4
3. The source code of your implementation containing any non-standard libraries

7 Assessment Criteria

Marking will follow the guidelines given in the school student handbook (see link in the next section).

The following issues will be considered:

- Adherence to the requirements
- Quality of the solution provided
- Code quality
- Examples
- Insights and analysis demonstrated in the report
- Extension activities completed, if any

Some guidelines are as follows. For a mark up to the band 11-13 you must complete Part 1. For a mark up to the band 14-16 you must complete part 1 and make an attempt to at least one heuristic-based search algorithm in part 2. To obtain marks on the band 17-18 or above, a robust implementation and evaluation of part 1 and part 2 is required, and may include extension elements as those suggested in Part 3. All parts are to be accompanied by an insightful report and good implementation quality.

8 Policies and Guidelines

8.1 Marking

See the standard mark descriptors in the School Student Handbook

`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_-Descriptors`

8.2 Lateness Penalty

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):

`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#latenesspenalties`

8.3 Good Academic Practice

The University policy on Good Academic Practice applies:

`https://www.st-andrews.ac.uk/students/rules/academicpractice/`

Alice Toniolo
(a.toniolo@st-andrews.ac.uk)
October 12, 2017