Foundations of Artificial Intelligence

M. Helmert, T. Keller
S. Eriksson
Spring Term 2020

University of Basel
Computer Science

Exercise Sheet 5

Due: April 1, 2020

Important: For submission, consult the rules at the end of the exercise. Non-adherence to the rules will lead to your submission not being corrected.

Exercise 5.1 (2 points)

Consider the following heuristic for the Elevators problem from Exercise 4.3 with passengers P:

$$h(s) = (\sum_{p \in P} \mathit{emb}(p)) + 10 \cdot (\max_{p \in P} |\mathit{floor}(p) - \mathit{goalfloor}(p)|)\,,$$

where $\mathit{floor}(p)$ denotes the floor of either p (if p is not in an elevator) or e (if p is in elevator e), $\mathit{goalfloor}(p)$ denotes the floor p tries to reach, and $\mathit{emb}(p)$ is defined as follows:

$$\mathit{emb}(p) = \begin{cases} 0 & \text{if } p \text{ is on their goal floor} \\ 1 & \text{if } p \text{ is in an elevator} \\ 2 & \text{if } p \text{ is on a floor other than their goal floor} \end{cases}$$

Determine if h is *safe*, *goal-aware*, *admissible* and/or *consistent*. Justify your answer for each property.

Exercise 5.2 (2+2 marks)

Show that the following two statements do not hold in general:

(a) If a heuristic is admissible, it is also consistent.

(b) If a heuristic is consistent, it is also admissible.

Hint: For each statement it suffices to present a counterexample consisting of a state space and a heuristic operating on this state space for which the left side of the statement holds, but the right side does not. In both cases counterexamples containing at most three states can be found.

Exercise 5.3 (1+4+1 marks)

The task in this exercise is to write a software program. We expect you to implement your code on your own, without using existing code you find online. If you encounter technical problems or have difficulties understanding the task, please let us know. For this exercise, you only have to change the files `ElevatorStateSpace.java` and `AstarSearch.java`.

(a) Implement the heuristic from Exercise 5.1. We have extended the interface `StateSpace` with a method that returns a heuristic value for the given state (the method is called `public int h(State s)`). The `ElevatorStateSpace` class already contains the skeleton of this method, and you should implement the heuristic in there.

(b) Implement A* without node reopening in a new file `AstarSearch.java`. Your class must inherit from `SearchAlgorithmBase`. Make sure that the value of the member variable `expandedNodes` is updated correctly.

(c) Test your implementation on the example problem instances provided in the tarball on the course website. Set a time limit of 10 minutes and a memory limit of 2 GB for each run. On Linux, you can set a time limit of 10 minutes with the command `ulimit -t 600`. Running your implementation on the first example instance with

```
java -Xmx2048M AstarSearch elevators elevators_inst_1
```

sets the memory limit to 2 GB. You are free to use higher memory limits. In any case, mention the limit in your solution.

Report runtime, number of node expansions, solution length and solution cost for all instances that can be solved within the given time and memory limits. For all other instances, report if the time or the memory limit was hit.

Submission rules:

- Create a single PDF file (ending .pdf) for all non-programming exercises. If you want to submit handwritten parts, include their scans in the single PDF. Put the names of all group members on top of the first page. Use page numbers or put your names on each page. Make sure your PDF prints on A4 (fits the page size).
- For programming exercises, create only those Java textfiles (ending .java) required by the exercise. Put your names in a comment on top of each file. Make sure your code compiles and test it!
- For the submission, you can either upload the single PDF or prepare a ZIP file (ending .zip, .tar.gz or .tgz; not .rar or anything else) containing the single PDF and the Java textfile(s) and nothing else. Please do not use subdirectories in the ZIP.
- Only upload one submission per group. Do not upload several versions, i.e., if you need to resubmit, use the same file name again so that the previous submission is overwritten.