Foundations of Artificial Intelligence

M. Helmert
S. Sievers
Spring Term 2023

University of Basel
Computer Science

Exercise Sheet 4

Due: March 26, 2023

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 4.1 (1+1 marks)

Consider the "missionaries and cannibals" problem from the lecture. As a quick reminder: states in this problem are triples $\langle m, c, b\rangle \in \{0,1,2,3\} \times \{0,1,2,3\} \times \{0,1\}$, where m gives the number of missionaries, c the number of cannibals and b the number of boats that are at the *left* river bank in the given state. Keep in mind that the boat can carry no more than two persons.
Let the initial state be $\langle 3,3,1\rangle$ and the goal states be $\{\langle 0,0,0\rangle, \langle 0,0,1\rangle\}$.

(a) Apply depth-bounded tree search with a depth bound of 2 to the initial state (i.e., simulate a call to `depth_bounded_search(`$\langle 3,3,1\rangle$`, 2)`). Successor nodes are generated by applying the following actions in order, ignoring the ones that are inapplicable in a given state: transport 2 missionaries, transport 1 missionary, transport 1 missionary and 1 cannibal, transport 1 cannibal, transport 2 cannibals. Make sure to ignore illegal states, i.e., states where cannibals outnumber missionaries on either river bank.

Specify the algorithm execution by providing, in a nested manner, with which arguments the function was called and what was returned. The structure should look like this:

```
depth_bounded_search(⟨3,3,1⟩, 2)
    depth_bounded_search(..., 1)
       further call or return
    etc.
    return none/solution
```

(b) What is the result of applying depth-first search (without depth bound) to this problem? Justify your answer.

Exercise 4.2 (1+1 marks)

Which of the search algorithms shown in slide 52 of the printout version of chapter 12 would you use to find optimal solutions for the following problems? For breadth-first and uniform cost also consider whether to use the tree or graph variant. Justify your answer in one to two sentences.

(a) Brute-forcing a password of unknown length, where each action (with cost 1) consists of adding a character to the string and the goal is reached as soon as we have the correct string (no enter key needed).

(b) route planning (as in finding the shortest path on a map)

Exercise 4.3 (2 marks)

We consider the Sokoban variant defined in Exercise 2.3., i.e., with costs of move actions equal 0 and costs of push actions equal 1. However, we do not stick with the particular instance shown there but consider a general problem on a grid of size $x \times y$ and with n boxes. In this problem, a state is defined as a tuple $\langle a, b_1, \ldots, b_n\rangle$ denoting the position a of the agent and the positions b_i of the n boxes. Recall that a position is a cell $c_{i,j}$ with coordinates (i, j) in the $x \times y$ grid. Further recall

that each box has its own goal position. Therefore, we can define a general goal condition through a tuple $\langle b_1^\star, \ldots, b_n^\star \rangle$ and the set of goal states as $S_\star = \{\langle a, b_1, \cdots b_n \rangle \in S \mid b_i = b_i^\star \text{ for all } 1 \leq i \leq n\}$.

Consider the heuristic h, defined as follows for some state $s = \langle a, b_1, \ldots, b_n \rangle$:

$$h(s) = \sum_{i=1}^{n} \mathrm{mh}(b_i, b_i^\star)$$

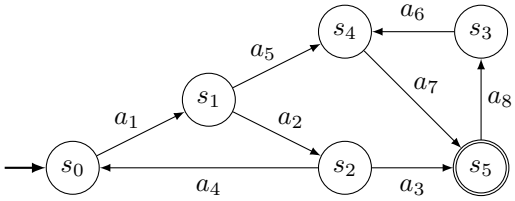
where the *Manhattan distance* between two cells (i.e., pairs of coordinates) is defined as $\mathrm{mh}(c_{i,j}, c_{i',j'})) = |i - i'| + |j - j'|$. In words, the heuristic computes the sum of the shortest "grid paths" (ignoring walls) of each box.

Formally prove that h is safe, goal-aware, admissible and consistent.

Hint: You may use results from the lecture slides. You may also use the fact that $\mathrm{mh}(c, x) - \mathrm{mh}(c', x) \leq 1$ *if* $\mathrm{mh}(c, c') = 1$.

Exercise 4.4 (2 marks)

For the state space depicted below with uniform action costs of 1, define a heuristic that is consistent, not safe and has a perfect estimate for s_0 (i.e., $h(s_0) = h^*(s_0)$). Justify your answer by showing why the heuristic is consistent and not safe.
Hint: If a consistent heuristic assigns ∞ *to a state, all its successors must be assigned* ∞ *as well.*



Exercise 4.5 (1+1 marks)

Execute

(a) A* ($f(n) = g(n) + h(n.\text{state})$) and

(b) greedy best-first search ($f(n) = h(n.\text{state})$)

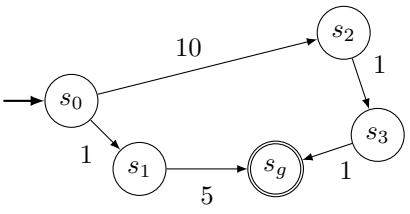
without reopening in the state space depicted below. As heuristic, use the perfect heuristic h^*. Describe the execution of both search algorithms with the following schema:

1. expanding s_{10}: *open* $= \langle s_{11}(f = 5), s_{12}(f = 6) \rangle$, *closed* $= \{s_{10}(g = 0)\}$
2. expanding s_{11}: *open* $= \langle s_{13}(f = 4), s_{12}(f = 6), s_{16}(f = 10) \rangle$, *closed* $= \{s_{10}(g = 0), s_{11}(g = 3)\}$

   ...

i. expanding s_i: found goal with cost x

Note: The schema above uses made up states and numbers that don't correspond to the given state space.



Submission rules:

- Exercise sheets must be submitted in groups of two students. Please submit a single copy of the exercises per group (only one member of the group does the submission).
- Create a single PDF file (ending .pdf) for all non-programming exercises. Use a file name that does not contain any spaces or special characters other than the underscore "_". If you want to submit handwritten solutions, include their scans in the single PDF. Make sure it is in a reasonable resolution so that it is readable, but ensure at the same time that the PDF size is not astronomically large. Put the names of all group members on top of the first page. Either use page numbers on all pages or put your names on each page. Make sure your PDF has size A4 (fits the page size if printed on A4).
- For programming exercises, only create those code textfiles required by the exercise. Put your names in a comment on top of each file. Make sure your code compiles and test it. Code that does not compile or which we cannot successfully execute will not be graded.
- For the submission: if the exercise sheet does not include programming exercises, simply upload the single PDF. If the exercise sheet includes programming exercises, upload a ZIP file (ending .zip, .tar.gz or .tgz; *not* .rar or anything else) containing the single PDF and the code textfile(s) and nothing else. Do not use directories within the ZIP, i.e., zip the files directly.
- Do not upload several versions to ADAM, i.e., if you need to resubmit, use the same file name again so that the previous submission is overwritten.