Foundations of Artificial Intelligence

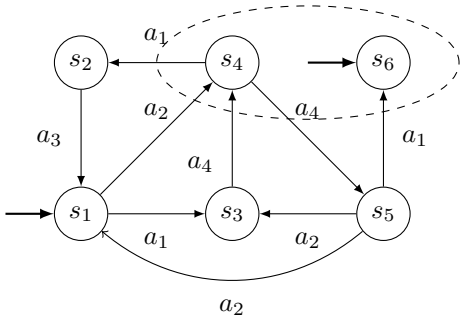
M. Helmert, T. Keller
S. Eriksson
Spring Term 2020

University of Basel
Computer Science

Presence Exercise 1

Exercise 1.1

Find all the flaws in the following graphical representation of a state space:



$cost : \{a_1 \mapsto 1, a_2 \mapsto 0, a_3 \mapsto -2, a_4 \mapsto 3.2\}$

Exercise 1.2

Determine if the following statements about state spaces $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$ are correct or not. Explain your answer.
Remark: The cardinality of a set X is denoted by $|X|$.

(a) An optimal solution is always a path of minimal length.

(b) If there is at least one state $s_\star \in S_\star$ that is reachable from s_0, then there is a solution for $\mathcal{S}$.

(c) There is a solution for $\mathcal{S}$ iff there is an optimal solution for $\mathcal{S}$.

(d) A successor state s' of a state s is always different from s.

(e) If all actions cost 0 there is a solution.

(f) Every action is applicable in at least one state.

(g) There are state spaces where $|A| < |T|$, there are state spaces where $|A| = |T|$ and there are state spaces where $|T| < |A|$.

(h) There is no solution for $\mathcal{S}$ if $T = \emptyset$.

(i) In a search space with no 0-cost actions, the length of an optimal solution is no larger than $|S| - 1$.

Exercise 1.3

This exercise is part of the next exercise sheet. The goal for this exercise session is mainly to get the program running and to detect any issues early on. We do not expect you to finish the exercise during this session.
The task in this exercise is to write a software program. We expect you to implement your code without using existing code you find online. If you encounter technical problems or have difficulties understanding the task, please let us know.
You can find sample Java code for the state space of the blocks world problem that was presented in the lecture on the website of the course. It implements the provided `State` and `Action` interfaces

as well as the black box interface for state spaces that was discussed in the lecture. Test the implementation by invoking the `StateSpaceTest` class, which creates a set of random successor states starting from the initial state. To run the program, first compile it with

```
javac StateSpaceTest.java
```

from a shell (on Linux) and then run it with

```
java StateSpaceTest blocks blocks-problem.txt
```

The sole purpose of the provided blocks world implementation is to serve as an example that helps you for your own implementation of an elevator dispatch problem in a new file called `ElevatorsStateSpace.java`.

(a) Implement the state space of an elevators dispatch problem where

- x passengers need to be transported by n elevators across m floors,
- the initial state describes on which floor each passenger is waiting and on which floor each elevator currently is,
- passengers can embark and disembark on an elevator, and an elevator can move one floor up or down at a time, and
- the goal is to deliver each passenger to their designated floor.

(b) Implement the `buildFromCmdline` function for the elevators dispatch problem such that it parses an input file with the following syntax:

- first line: total number of passengers, elevators and floors (each separated by a whitespace).
- second line: one integer for each passenger (in order, separated by a whitespace) indicating the floor they are waiting on
- third line: one integer for each elevator (in order, separated by a whitespace) indicating the floor they start at
- fourth line: one integer for each passenger (in order, separated by a whitespace) indicating their destination floor
- note: floors are numbered from 0 to $m - 1$

You can find the sample input file `elevators-problem.txt` on the website of the course. It encodes an instance with 3 passengers, 2 elevators and 5 floors (floor 0 to floor 4). The first passenger waits on foor 3, the second on floor 1 and the third on floor 4. The two elevators start at floor 0 and floor 1 respectively. The goal is to transport the first passenger to floor 2, the second to floor 4 and the third to floor 0.

To test your implementation, uncomment the two indicated lines in the method `createStateSpace` of the `StateSpaceTest` class. You can then execute the command

```
java StateSpaceTest elevators elevators-problem.txt
```

Notice that the command uses the new keyword `elevators` (rather than `blocks` as in the example above).