Foundations of Artificial Intelligence

M. Helmert
J. Seipp
Spring Term 2019

University of Basel
Computer Science

Exercise Sheet 2

Due: March 6, 2019

Exercise 2.1 (1+1 marks)

Characterize the following environments by describing if they are *static* / *dynamic*, *deterministic* / *non-deterministic* / *stochastic*, *fully* / *partially* / *not observable*, *discrete* / *continuous*, and *single-agent* / *multi-agent*. Explain your answer.

(a) Sokoban (see, e.g., `https://en.wikipedia.org/wiki/Sokoban`)

(b) Monopoly (see, e.g., `https://en.wikipedia.org/wiki/Monopoly_(game)`)

Exercise 2.2 (0.5 + 0.5 + 0.5 + 0.5 + 0.5 + 0.5 marks)

Determine if the following statements are true for all state spaces $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$. Explain your answers. Remark: the cardinality of a set X is denoted by $|X|$.

(a) If all actions have equal cost, each solution for $\mathcal{S}$ is optimal.

(b) There can be infinitely many transitions in T.

(c) The length of an optimal solution is no larger than $|S| - 1$.

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

(e) If $\pi = \langle \pi_1, \dots, \pi_n \rangle$ is a minimal cost path from state $s \in S$ to state $s' \in S$ and $\pi' = \langle \pi'_1, \dots, \pi'_m \rangle$ is a minimal cost path from s' to state $s'' \in S$, then $\pi'' = \langle \pi_1, \dots, \pi_n, \pi'_1, \dots, \pi'_m \rangle$ is a minimal cost path from s to s''.

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

(g) Let π, π', and π'' be minimal cost paths from $s \in S$ to $s' \in S$, s' to $s'' \in S$ and s to s'', respectively. Then $cost(\pi'') \leq cost(\pi) + cost(\pi')$.

Exercise 2.3 (2+1 marks)

(a) Formalize the state space of the 15-puzzle that was introduced in Chapter 5 of the lecture. Specify the set of states, the set of actions, the action costs, and the set of transitions. Provide the initial state and the set of goal states as depicted on slide 5 of the print version of Chapter 5.

(b) How many states are in the set of states of the 15-puzzle? How many states are in the set of goal states?

Exercise 2.4 (4 marks)

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 a *restricted* blocks world variant in a new file called `RestrictedBlocksStateSpace.java`.

- Implement the state space of a blocks world variant where
  - the maximum number of table positions (and hence towers) is limited by a constant
  - the maximal height of each tower is limited by a constant (separately for each table position)
  - the goal is to create towers of specific blocks in specific table positions
- Implement the `buildFromCmdline` function for the blocks world variant such that it parses an input file with the following syntax:
  - first line: total number of blocks and maximum number of table positions
  - one line for each table position: maximum number of blocks in the tower at this table position, followed by the IDs of the blocks in the initial state, starting from the surface of the table (end with `-1`)
  - one line for each table position: IDs of the blocks in the goal state, starting from the surface of the table (end with `-1`)

  You can find the sample input file `restricted-blocks-problem.txt` on the website of the course. It encodes an instance with 4 blocks and 3 table positions. The tower in the first table position has a maximum height of 1 and is empty initially, the tower in the second table position has a maximum height of 4 and contains the tower of blocks with IDs 1, 2, and 4 and the tower in the third table position has a maximum height of 2 and contains a tower that consists only of block 3. The goal is to have a tower of blocks with increasing IDs in the second table position.

You should only create a single new file `RestrictedBlocksStateSpace.java` and uncomment the two indicated lines in the method `createStateSpace` of the `StateSpaceTest` class. In particular, you should not create or change other files.
To test your implementation, you can execute the command

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

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

Important: Solutions should be submitted in groups of two students. However, only one student should upload the solution. Please provide both student names on each file you submit. We can only accept a single PDF or a ZIP file containing *.java or *.pddl files and a single PDF.