Planning and Optimization

M. Helmert, G. Röger
P. Ferber, T. Keller, S. Sievers

University of Basel
Fall Semester 2020

Classroom Exercise 1

The files required for this exercise are in the directory `classroom-exercise-1` *of the course repository (*`/vagrant/planopt-hs20` *in your course VM). Update your clone of the repository with* `git pull` *to see the files. For the runs with Fast Downward, set a time limit of 1 minute and a memory limit of 2 GB. Using Linux, such limits can be set with* `ulimit -t 60` *and* `ulimit -v 2000000`*, respectively.*

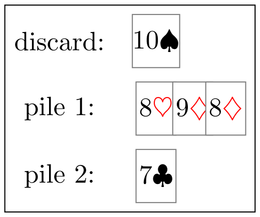
Exercise 1 (Mathematical Modeling)

In this exercise, you must use functions, sets, and tuples to formalize the following situations in card games:

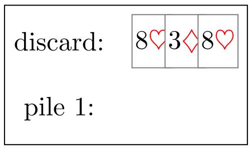
- There is a set of suits and a set of (integer) values.
- Each card has a suit and a value.
- There are some tableau piles of cards with at most one card on top of each card and one discard pile.
- If successive cards on a tableau pile have the same suit, their values must be decreasing.

(a) Define a general form that fits all such situations by using variables for the specific cards and piles. For example, use a variable *Cards* for the set of all cards that occur in the all piles. Any combination of values for the variables that match your definition should describe a situation according to the description above and any situation should be expressible as a combination of values.

(b) Show how the following specific situation fits your general form.



(c) Show how the following specific situation fits your general form.



Exercise 2 (Running Fast Downward)
Play around with the Fast Downward planner:

(a) The directory `classroom-exercise-1/tile` contains two variants of the 15-Puzzle:

- original formulation (`puzzle.pddl`, `puzzle01.pddl`)
- variant with weighted tiles (`weight.pddl`, `weight01.pddl`)

To run Fast Downward, use the script `fast-downward.py` with the corresponding domain and problem files, specifying the search algorithm and the heuristic. Example for the original formulation, greedy best first search and the FF heuristic:

```
./fast-downward/fast-downward.py tile/puzzle.pddl tile/puzzle01.pddl \
          --heuristic "h=ff()" --search "eager_greedy([h])"
```

Run Fast Downward on the 15-Puzzle and the Weighted 15-Puzzle, using greedy best first search and different heuristics:

- blind heuristic: `blind()`
- additive heuristic: `add()`
- FF heuristic: `ff()`

(b) Compare the results with respect to time, number of expanded and generated states, and solution quality.

Exercise 3 (Package Delivery)
Consider the following package delivery problem:

- There is a set of *cities*, *trucks*, and *packages*.
- The cities are *connected* by a road network.
- A truck can *move* from one city to another along the roads.
- A package can be *loaded* into and *unloaded* from a truck.
- Every package has a *target location* where it should be delivered.

Model this domain in PDDL. Then model at least two different instances (e.g. different road networks, different target locations, different number of trucks) and find plans for them with Fast Downward.