## Planning and Optimization

M. Helmert, G. Röger
T. Keller, M. Wehrle

University of Basel
Fall Semester 2016

# Exercise Sheet 1

**Due: October 5, 2016**

*For the runs with Fast Downward, set a time limit of 10 minutes and a memory limit of 2 GB. Using Linux, such limits can be set with* `ulimit -t 600` *and* `ulimit -v 2000000`*, respectively.*

**Exercise 1.1** (2+3+3+4 marks)

The *Knight's Tour Problem* (*Rösslesprung-Problem*) is defined as follows.

- Given a chess board, which is empty except for a single knight $K$,
- find a sequence of moves such that every square is visited by $K$ exactly once.

According to the rules of chess, the knight can move to any square that is two squares horizontally and one square vertically, or two squares vertically and one square horizontally. On the course website, you can find a tarball (`knights_tour.tar`), containing the domain file `knights_tour.pddl` and problem files for a Knight's Tour problem on a $5 \times 5$ board (`knights_tour5.pddl`) and on an $8 \times 8$ board (`knights_tour8.pddl`).

a) Run Fast Downward on `knights_tour5.pddl` and `knights_tour8.pddl`, using $A^*$ search and the canonical PDB heuristic (`h=cpdbs()`), the maximum heuristic (`h=hmax()`), and the blind heuristic (`h=blind()`). Compare the results with respect to time, number of expanded states, and plan costs. Recall that Fast Downward with $A^*$ is executed as:

```
./fast-downward.py <domain.pddl> <problem.pddl> --heuristic "h=<heuristic>"
                                                --search "astar(h)"
```

b) Create a domain `knights_tour_blocked.pddl` which differs from `knights_tour.pddl` only in the fact that squares on the board can be *blocked* in a way that the knight cannot jump on blocked squares. Based on `knights_tour8.pddl`, generate two problem files named `knights_tour8_blocked_1.pddl` and `knights_tour8_blocked_2.pddl` so that $h1$ is blocked in the former, and $a1, b1, c1, d1, f1, g1$, and $h1$ are blocked in the latter. You also have to modify the goal state such that only unblocked squares must be visited. Run Fast Downward with $A^*$ and `h=cpdbs()` on the modified problems. Compare and discuss the results with the results from a) with respect to time, number of expanded states, and plan costs.

c) Blocking squares (and requiring that only unblocked squares need to be visited in the goal) can yield unsolvable problems. Identify a criterion to generate unsolvable problems. Justify your answer. Create an unsovable problem `knights_tour8_blocked_3.pddl` and apply Fast Downward ($A^*$, `h=cpdbs()`). Discuss the number of expanded states and the search time.

d) Consider the *Duplicate Knight's Tour problem* where the knight can additionally duplicate itself once while moving. For this, there is an additional action `duplicate(?from ?to)` that differs from the regular move action by simultaneously moving the knight to `?to` while still keeping the knight at `?from`. In other words, the knight is at both squares simultaneously afterwards. The duplicate action is applicable at most once. Implement the Duplicate Knight's Tour problem in `knights_tour_duplicate.pddl` and `knights_tour8_duplicate.pddl` by adjusting `knights_tour.pddl` and `knights_tour8.pddl` accordingly. Run Fast Downward on the problem, using $A^*$ and the heuristics from part a). Discuss the results with respect to time, number of expanded states, and plan costs compared to the results from part a).

*For parts b), c) and d), please also submit the modified PDDL files.*

*The exercise sheets can be submitted in groups of two students. Please provide both student names on the submission.*