Planning and Optimization

M. Helmert, G. Röger
R. Christen, P. Ferber, T. Keller

University of Basel
Fall Semester 2022

Exercise Sheet 6

Due: November 07, 2022

Important: for submission, consult the rules at the end of the exercise. Non-adherence to these rules might lead to a penalty in the form of a deduction of marks or, in the worst case, that your submission will not be corrected at all.

Exercise 6.1 (1.5+0.5+0.5+0.5 marks)

Consider the planning task $\Pi = \langle V, I, O, \gamma \rangle$ in positive normal form with

$$
\begin{aligned}
V &= \{a, b, c, d\} \\
I &= \{v \mapsto \mathbf{F} \mid v \in V\} \\
O &= \{o_1, o_2, o_3, o_4\} \text{ where} \\
o_1 &= \langle \top, a \wedge (a \rhd b), 3 \rangle \\
o_2 &= \langle a, c, 6 \rangle \\
o_3 &= \langle a \wedge b, c, 1 \rangle \\
o_4 &= \langle b \vee c, \neg c \wedge d, 2 \rangle \\
\gamma &= c \wedge d
\end{aligned}
$$

(a) Provide the relaxed task graph for Π.

Hint: We ask you to add annotations to your solution in parts (b) to (d), so please make sure to leave enough space. Drawing some operator nodes above and some below the variable nodes yields a much clearer solution with more space for annotations.

(b) Annotate your relaxed task graph from part (a) with the costs that are computed for $h^{\max}$. What is $h^{\max}(I)$?

(c) Annotate your relaxed task graph from part (a) with the costs that are computed for h^{add}. What is $h^{\text{add}}(I)$?

(d) Mark all best achievers in your relaxed task graph from part (a). What is $h^{\text{FF}}(I)$?

Exercise 6.2 (0.5+0.5+0.5+0.5 marks)

(a) Give a unit-cost STRIPS planning task $\Pi_1 = \langle V_1, I_1, O_1, \gamma_1 \rangle$ with at most 2 state variables and an initial state I_1 with $h^{\text{add}}(I_1) > h^{\text{FF}}(I_1)$ or explain why this is not possible.

(b) Give a unit-cost STRIPS planning task $\Pi_2 = \langle V_2, I_2, O_2, \gamma_2 \rangle$ with at most 2 state variables and an initial state I_2 with $h^{\max}(I_2) < h^{+}(I_2)$ or explain why this is not possible.

(c) Give a unit-cost STRIPS planning task $\Pi_3 = \langle V_3, I_3, O_3, \gamma_3 \rangle$ with at most 2 state variables and an initial state I_3 with $h^{+}(I_3) = \infty$ and $h^{*}(I_3) < \infty$ or explain why this is not possible.

(d) Which of the heuristics h^{+}, $h^{\max}$, h^{add} and h^{FF} would you recommend for an optimal planning algorithm? Justify why the heuristic you choose is the most appropriate one.

Exercise 6.3 (2+2+1 marks)

In this exercise, we ask you to implement an algorithm that is presented in the lecture. You may assume that your implementation is called with a STRIPS planning task. When you execute your code, use a time limit of 30 seconds, which can be imposed by executing `ulimit -t 30` in the console after logging into the virtual machine. Use the benchmark instances of the SOKOBAN domain for your experiments. They can be found in the directory `benchmarks/sokoban`.

(a) The file `fast-downward/src/search/planopt_heuristics/relaxed_task_graph.cc` contains a partial implementation of a relaxed task graph for STRIPS tasks. Complete it by constructing the appropriate AND/OR nodes and edges between them in the constructor. Associate operator effect nodes with costs (only needed for later parts of this exercise).

To test your implementation, you can use the heuristic `planopt_relaxed_task_graph()` (which prunes states that are not relaxed solvable). The plan costs of the first two instances of the SOKOBAN *domain are 32 and 10, respectively.*

(b) Implement the method `ff_cost_of_goal` by collecting all best achievers. Start from the goal node and recursively collect all successors of each encountered AND node and the stored best achiever from each encountered OR node. Return the sum of direct costs of all collected nodes.

To test your implementation, you can use a greedy best-first search with your heuristic by invoking Fast Downward with `--search "eager_greedy([planopt_ff()])"`. The plan costs of the first two instances of the SOKOBAN *domain are 32 and 10, respectively. You can also compare the values of `planopt_ff()` to those of the built-in implementation of Fast Downward (`ff()`). The values are not guaranteed to match, but should lead to similar results on the* SOKOBAN *benchmarks.*

(c) Evaluate the heuristic from part (b) in a greedy best-first search on the SOKOBAN instances. Compare the following values: $h^{\text{add}}(I)$, $h^{\text{FF}}(I)$, $h^{+}(I)$, $h^{*}(I)$ and $c(\pi)$, where π is the plan computed with h^{FF}.

Fill in the table below, then discuss the results, in particular w.r.t. the relationship of the different heuristic values/plan costs.

Instance	$h^{\text{add}}(I)$	$h^{\text{FF}}(I)$	$h^{+}(I)$	$h^{*}(I)$	$c(\pi)$
p01					
p02					
p03					
p04					
p05					
p06					
p07					
p08					

You can compute $h^{\text{add}}(I)$ by using the heuristic `planopt_add()` with any search engine, e.g `--search "astar([planopt_add()])"`. You can compute optimal plans by running $A^{\star}$ with any admissible heuristic (e.g. with `--search "astar(lmcut())"`) and optimal relaxed plans by explicitly creating the delete relaxation of the task and solving it with an optimal search algorithm (e.g., with `--search "astar(lmcut())" --translate-options --relaxed`). This is not the ideal way of computing optimal relaxed plans, so it will not

complete on all instances. If the search does not complete in 30 seconds, the last reached f-layer is a lower bound to the optimal relaxed solution cost. In these cases, please provide the value x of the last reached f-layer and write an table entry of the form $\geq x$.

Submission rules:

- Exercise sheets must be submitted in groups of two or three 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. After creating your zip file and before submitting it, open the file and verify that it complies with these requirements.
- 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.