Planning and Optimization

M. Helmert, G. Röger
C. Büchner, R. Christen, S. Dold

University of Basel
Fall Semester 2022

Exercise Sheet 5

Due: November 6, 2023

Important: This exercise sheet is twice the usual workload: It covers two weeks of lectures, the time between publication and due date spans two weeks, and the maximal number of marks is 20 instead of the usual 10.
Note: At the time of publishing these exercises, not all topics relevant for solving them were already discussed in the lecture. Consider to start working on all individual exercises nevertheless, because the order of the exercises does not strictly follow the course schedule. In particular, exercise 5.3, which is a programming exercise, has some parts that are covered in the first week of lectures and others that are covered only in the second week of lectures on delete relaxation.

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 marked at all.

Exercise 5.1 (0.5+0.5+0.5+0.5+0.5+0.5 marks)
Consider the STRIPS planning task $\Pi = \langle V, I, O, \gamma \rangle$ with

- $V = \{a, b, c, d, e\}$;
- $I = \{a \mapsto \mathbf{T}, b \mapsto \mathbf{F}, c \mapsto \mathbf{F}, d \mapsto \mathbf{F}, e \mapsto \mathbf{F}\}$;
- $O = \{o_1, o_2, o_3, o_4\}$ where
  - $o_1 = \langle a, b \wedge c, 1 \rangle$,
  - $o_2 = \langle b \wedge c, d, 1 \rangle$,
  - $o_3 = \langle c, \neg c \wedge e, 1 \rangle$, and
  - $o_4 = \langle d, \neg a \wedge e, 1 \rangle$; and
- $\gamma = c \wedge e$.

(a) What is the on-set of $I[\![o_1]\!]$?

(b) Provide a state $s \neq I$ reachable from I that dominates I.

(c) Provide a state s' reachable from s that does not dominate s.

(d) Provide an optimal plan for Π.

(e) Provide the delete relaxation Π^+ of Π.

(f) Provide an optimal *relaxed* plan for Π, i.e., a plan for Π^+.

Exercise 5.2 (0.5+1.5+0.5+0.5 marks)
Consider the propositional planning task $\Pi = \langle V, I, O, \gamma \rangle$ with

- $V = \{a, b, c, d, e, f\}$;
- $I = \{a \mapsto \mathbf{T}, b \mapsto \mathbf{F}, c \mapsto \mathbf{T}, d \mapsto \mathbf{F}, e \mapsto \mathbf{F}, f \mapsto \mathbf{F}\}$;
- $O = \{o_1, o_2, o_3\}$ where
  - $o_1 = \langle (a \vee b) \wedge c, d, 1 \rangle$,
  - $o_2 = \langle \top, b \wedge (d \rhd e), 1 \rangle$, and
  - $o_3 = \langle e, \neg c \wedge f, 1 \rangle$; and
- $\gamma = c \wedge f$.

(a) Is Π in positive normal form? Justify your answer.

(b) Visualize the relaxed task graph for Π.

(c) Is Π relaxed solvable? Justify your answer.

(d) Is Π solvable? Justify your answer.

Exercise 5.3 (3+2+3+2 marks)
Update the course repository (`/vagrant/planopt-hs23` in your course VM) with `$ git pull`. Navigate to the new directory `sheet05` which contains the files required for this exercise. All files you need to change are in the directory `fast-downward/src/search/planopt_heuristics/`.

For this exercise you may assume that the problems used for testing are STRIPS planning tasks. When you execute your code, use a time limit of 2 minutes, which can be imposed by executing `ulimit -S -t 120` in the console after logging into the virtual machine. Use the benchmark instances in the directory `castle`.

(a) The files `and_or_graph.*` contain an implementation of an AND/OR graph. Implement the method `most_conservative_valuation` as outlined in the code comments to find the most conservative valuation of a given AND/OR graph.

Hint: The method `test_an_or_graphs` implements the example graphs from the lecture. You can use them to test and debug your implementation by calling Fast Downward as `./fast-downward.py --test-and-or-graphs`.

(b) Implement the method `is_goal_relaxed_reachable` in `relaxed_task_graph.cc`. Then use the heuristic `planopt_relaxed_task_graph()` which prunes states that are not relaxed solvable based on the result of that function. Use an A* search on the instances in the directory `castle` and compare the results to blind search (heuristic `blind()`) with respect to the number and speed of expansions.

(c) Implement the method `weighted_most_conservative_valuation` for AND/OR graphs to compute h^{add} by following the approach outlined in the code comments. Use a comment to point out the change you would have to make to turn this into a computation for $h^{\max}$.

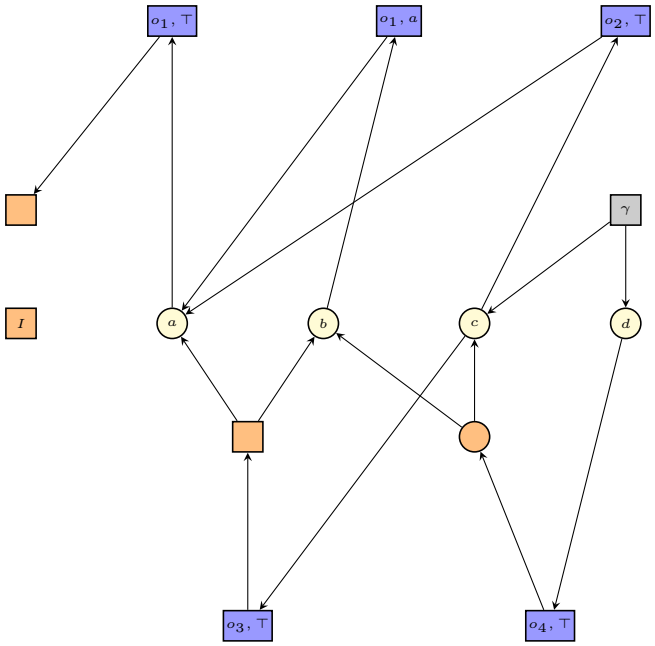
(d) Implement the method `additive_cost_of_goal` in `relaxed_task_graph.cc`. It should return the h^{add} value of the task based on the implementation of part (c). Then use the heuristic `planopt_add()` in an eager greedy search on the instances in the directory `castle` and compare the heuristic values of the initial state with the cost of an optimal relaxed plan, the discovered plan and an optimal plan.

Hint: The values of `planopt_add()` and the built-in implementation of Fast Downward (`add()`) should match, so you can use the built-in implementation for debugging part (c).

Exercise 5.4 (1+1+1 marks)
Consider the propositional planning task $\Pi = \langle V, I, O, \gamma \rangle$ in positive normal form with

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

The relaxed task graph for Π looks as follows:



(a) Annotate to the left of each node its $h^{\max}$ cost. What is $h^{\max}(I)$?

(b) Annotate to the right of each node its h^{add} cost. What is $h^{\text{add}}(I)$?

(c) Mark all best achievers in the relaxed task graph of Π. What is $h^{\text{FF}}(I)$?

Exercise 5.5 (1 mark)
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.

Submission rules:

- Exercise sheets must be submitted in groups of 2–3 students. Create a team on ADAM including all members of your group and submit a single copy of the exercises per group.
- 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 text file(s) 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 text file(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.