

# Planning and Optimization

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

University of Basel  
 Fall Semester 2022

## Exercise Sheet 5 Due: October 31, 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 5.1 (2+0.5+0.5 marks)

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

$$\begin{aligned} V &= \{a, b, c\} \\ I &= \{a \mapsto \mathbf{T}, b \mapsto \mathbf{F}, c \mapsto \mathbf{T}\} \\ O &= \{o_1, o_2, o_3\} \\ o_1 &= \langle b, c \rangle \\ o_2 &= \langle \neg c, \neg a \wedge b \rangle \\ o_3 &= \langle \top, b \wedge (\neg b \triangleright \neg c) \rangle \\ \gamma &= \neg a \wedge c \end{aligned}$$

- (a) Transform  $\Pi$  to a propositional planning task  $\Pi'$  in positive normal form and provide the delete relaxation  $\Pi^+$  of  $\Pi'$ .
- (b) Provide optimal plans for  $\Pi$ ,  $\Pi'$  and  $\Pi^+$ .
- (c) Provide the on-sets  $\text{on}(I' \llbracket o_3 \rrbracket)$  and  $\text{on}(I' \llbracket o_3^+ \rrbracket)$ .

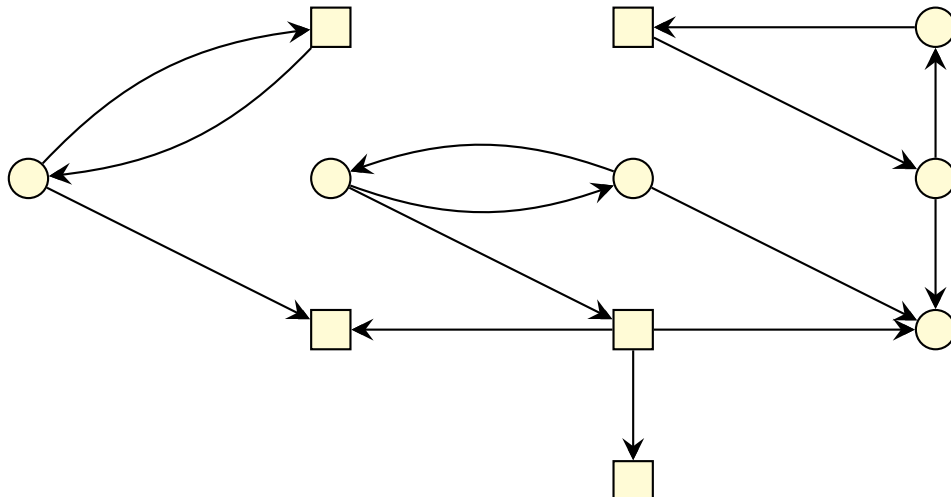
### Exercise 5.2 (2 marks)

Consider a planning task  $\Pi = \langle V, I, O, \gamma \rangle$  in positive normal form, a goal state  $s$  of  $\Pi$  and an operator  $o$  which is applicable in  $s$ .

Show that  $s \llbracket o^+ \rrbracket \models \gamma$  (using the lemmata from the lecture) and provide an example where  $s \llbracket o \rrbracket \not\models \gamma$ .

### Exercise 5.3 (0.5+0.5+0.5+0.5 marks)

Consider the following AND/OR graph:



Use the four smaller versions of this AND/OR graph below to annotate the nodes with truth values **T** and **F** as given by the following:

- Mark all forced true and forced false nodes.
- Mark all nodes with their truth value in the most conservative valuation.
- Mark all nodes with their truth value in the least conservative valuation.
- Mark all nodes with their truth value in a consistent valuation that is different from the most conservative and the least conservative valuation.

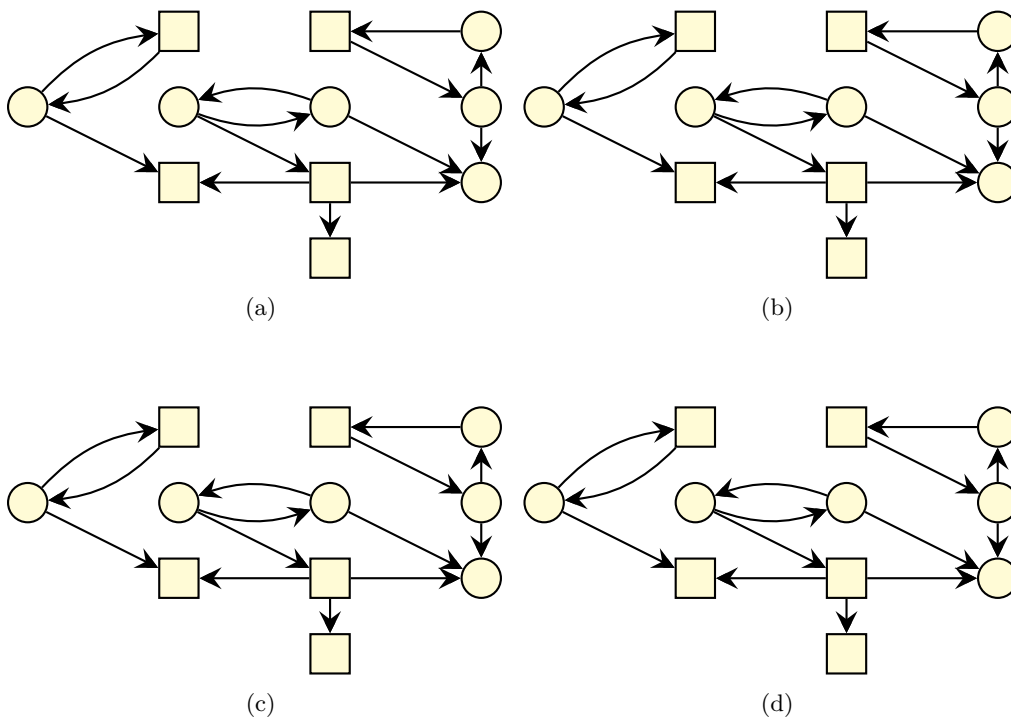


Figure 1: Graphs to annotate for exercise 5.2.

**Exercise 5.4** (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 CASTLE domain for your experiments. They can be found in the directory `benchmarks/castle`.

- In the file `fast-downward/src/search/planopt_heuristics/h_greedy_relaxed_plan.cc` you can find an incomplete implementation of the greedy algorithm for relaxed planning tasks that has been presented in the lecture. Complete the implementation.

*To test your implementation, you can use a greedy best-first search with your heuristic by invoking Fast Downward with `--search "eager-greedy([planopt-greedy-relaxed()])"`. The plan costs of the first two instances of the CASTLE domain are 4 and 16, respectively. Note that the order in which operators are considered during the computation of the heuristic matters, so your implementation could be correct despite obtaining different values.*

- (b) Evaluate the heuristic from part (a) in a greedy best-first search on the CASTLE instances. Compare the heuristic values of the initial state ( $h^{\text{greedy}}(I)$ ) with the cost of an optimal relaxed plan ( $h^+(I)$ ), an optimal plan ( $h^*(I)$ ) and the cost of the discovered plan using  $h^{\text{greedy}}$  ( $c(\pi)$ ).

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

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

*You can compute optimal plans by running A\* 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.