Automated Planning (TDDD48)

Jendrik Seipp
Mika Skjelnes

Linköping University

Lab 4

Important: For submission, consult the rules at the end of this document. Non-adherence to these rules might lead to a penalty in the form of a deduction of points. Some points are *bonus points*. These can help you reach the point quota per lab (4/12 points) and the overall point quota ($50\% \cdot 7 \cdot 12 = 42$ points).

Exercise 4.1 (0.5+1.5+0.5+0.5 points)
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.

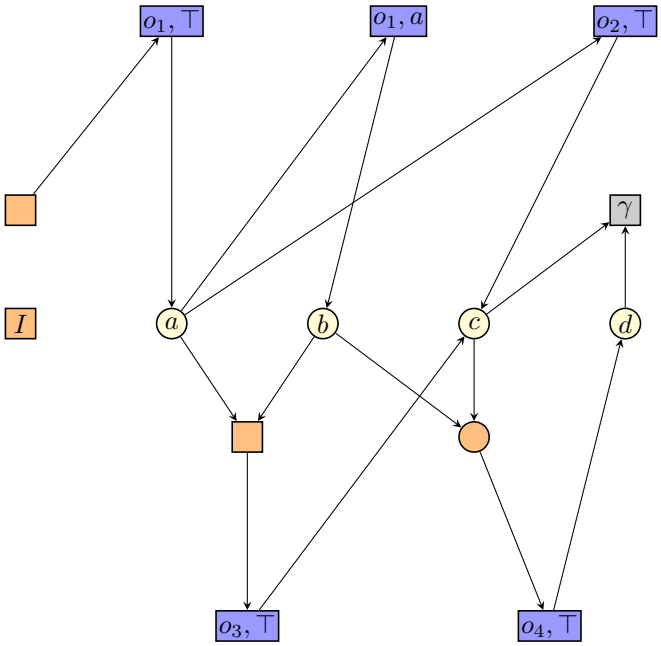
(c) Is Π relaxed solvable? Justify your answer.

(d) Is Π solvable? Justify your answer.

Exercise 4.2 (1+1+1 points)
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 4.3 (4+2 points)

Update the course repository (`/vagrant/tddd48` in your course VM) with `git pull`. Navigate to the new directory `lab4` which contains the files required for this exercise.
In this exercise, your task is to implement the greedy algorithm for relaxed planning tasks. 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` directory for your experiments.

(a) 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. 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()])"`. *You find all relevant information for this exercise in the header files in the planopt_heuristics directory. You should not have to inspect any other files.*

(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^{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 $A^\star$ 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 $\geq x$.

Bonus Exercise 4.4 (0.5+0.5+2+1=4 bonus points)

In the *pancake problem* we have n pancakes with size $1, \ldots, n$ on a pile. The goal is to order the pile by size, i.e., the largest pancake is on the bottom, the second largest on top of the largest and so on. The pile can be manipulated by removing a stack of i pancakes from the top and putting it back in reverse order. Hence, there is an action flip_i for all $1 \leq i \leq n$. The following figure shows an example instance:

initial state | apply flip_4 to initial state | goal state

Let $P = \{p_1, p_2, p_3, p_4, p_5\}$ be the set of pancakes where p_1 is the smallest pancake and p_5 is the largest, and let $N = \{1, 2, 3, 4, 5\}$ be the set of positions in the pile where 1 is on top and 5 is at the bottom. Consider the following formalization of the above instance of the pancake problem as a propositional planning task $\Pi = \langle V, I, O, \gamma \rangle$:

- $V = \{p_i\texttt{-at-}j \mid p_i \in P, j \in N\}$;
- $I = \{v \mapsto \mathbf{T} \mid v \in V_I\} \cup \{v \mapsto \mathbf{F} \mid v \in V \setminus V_I\}$ with
  $V_I = \{p_1\texttt{-at-}4, p_2\texttt{-at-}2, p_3\texttt{-at-}1, p_4\texttt{-at-}5, p_5\texttt{-at-}3\}$;
- $O = \{\mathit{flip}_i \mid 1 \leq i \leq 5\}$ with $\mathit{flip}_i = \langle \top, \mathit{eff}_i, 1 \rangle$ such that

$$\mathit{eff}_i = \bigwedge_{p \in P} \bigwedge_{j=1}^{i} (p\texttt{-at-}j \rhd (\neg p\texttt{-at-}j \wedge p\texttt{-at-}k))$$

  where $k = i - j + 1$; and
- $\gamma = \bigwedge_{i=1}^{5} p_i\texttt{-at-}i$.

(a) Provide the invariant stating that at most one of p_1, p_2 and p_3 may be on the bottom of the pile at once.

Note: There is no good reason to leave out p_4 and p_5 from this invariant apart from reducing the effort needed to solve this exercise.

(b) Provide a mutex group stating that a certain pancake cannot be at multiple positions in the pile at once.

(c) Formalize the above pancake instance as a planning task $\Pi' = \langle V', I', O', \gamma' \rangle$ in finite-domain representation such that V' contains exactly one variable per pancake (i.e., $|V'| = 5$).

(d) Compare the two planning tasks Π and Π'. How many states does each of them have? How many of those states are reachable?

Submission rules:

- Lab sheets must be submitted in groups of 2–3 students. Clone the labs repo (`https://github.com/mrlab-ai/tddd48-labs`) and push it to a repo at the University GitLab instance `https://gitlab.liu.se`. Make sure the repo is **private** and give read access to Mika Skjelnes (`mika.skjelnes@liu.se`).
- For non-programming exercises, create a single PDF file at the location `labX/solution.pdf`. 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. 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, directly edit the code in the cloned repository and only create those code text file(s) required by the lab. 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.