Planning and Optimization

M. Helmert, T. Keller
S. Eriksson, F. Pommerening, S. Sievers

University of Basel
Fall Semester 2019

Exercise Sheet E

Due: December 1, 2019

The files required for this exercise are in the directory `exercise-e` *of the course repository* (`https://bitbucket.org/aibasel/planopt-hs19`)*. All paths are relative to this directory. Update your clone of the repository with* `hg pull -u` *to see the files.*

Exercise E.1 (1+2+2 marks)

Consider the STRIPS task $\Pi = \langle V, I, O, \gamma \rangle$ with variable set $V = \{a, b, c, d, e, f, g\}$, initial state I with $I(a) = \top$ and $I(v) = \bot$ for all $v \in V \setminus \{a\}$, operator set $O = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ and goal $\gamma = e$. All operators in O have cost 1 and are defined as follows:

$$
\begin{aligned}
o_1 &= \langle a, b \wedge \neg a \rangle \\
o_2 &= \langle a, c \rangle \\
o_3 &= \langle b, d \rangle \\
o_4 &= \langle c, d \rangle \\
o_5 &= \langle d \wedge g, e \wedge f \rangle \\
o_6 &= \langle \top, g \rangle
\end{aligned}
$$

(a) Provide the simplified relaxed task graph $sRTG(\Pi^+)$ in graphical form.

(b) Compute the set of causal fact landmarks with the fixed-point algorithm introduced in chapter E2. You can annotate the nodes of your graph from (a) as in the lecture, but denote in which order you update the values of the nodes, and if you change the same node several times provide all intermediate values.

(c) Are the following two operator sets disjunctive action landmarks? Briefly justify your answer (no formal proof required):

- $O_1 = \{o_1, o_3\}$
- $O_2 = \{o_3, o_4\}$

Exercise E.2 (4+3+3+2 marks)

(a) In `fast-downward/src/search/planopt_heuristics/justification_graph.*`, you can find an incomplete implementation of the justification graph used by the LM-cut heuristic. Complete the implementation of the constructor and the methods `mark_goal_zone` and `find_cut_edges` by following the comments in the code. Then compute the LM-cut algorithm in the method `compute_heuristic` of the file `lmcut.cc` in the same directory. What is the heuristic value of the initial state of `tasks/exercise2a.pddl`? What is the number of expanded states excluding the last f-layer (printed as "Expanded until last jump") in the same task, if you run an A* search with your implementation of $h^{\text{LM-cut}}$?

The $h^{\max}$ *computation is integrated into the class* `DeleteRelaxedNormalFormTask` *for efficiency. The classes for operators and propositions track their* $h^{\max}$ *achievers and costs. Calling the method* `compute_hmax` *on the task will update these stored values.*

You can call your heuristic as `planopt_lmcut()`*. If you compare it to the built-in implementation, note that the result of each heuristic evaluation depends on the precondition-choice function that was used. The two implementations are not guaranteed to pick the same achievers, so they might give different results. However, in most tasks the heuristic value of the initial state should be similar.*

(b) Draw the justification graphs generated by LM-cut in the heuristic computation for the initial state of `tasks/exercise2b.pddl`. Label each node of the graph with the name of the represented proposition and its $h^{\max}$ value. Label transitions with the correct operator and its current cost. Also mark the goal zone and the cut in each graph.

You can do this exercise manually but it may be easier to adapt your algorithm from exercise (a) to print the necessary information during the computation.

(c) In the files `fast-downward/src/search/planopt_heuristics/and_or_graph.*` you can find an incomplete implementation of the algorithm discovering landmarks in AND/OR graphs. Complete the method `compute_landmarks` to compute the set of landmarks for reaching the given node in the graph.

You can use your implementation to compute disjunctive action landmarks and causal fact landmarks for I in sRTG(Π^+) by calling the planner as `./fast-downward.py /path/to/task --compute-landmarks`. The output lists all disjunctive action landmarks and the causal fact landmarks that are not already true in the initial state.

(d) Use your implementation from exercise (c) to compute action and fact landmarks for `tasks/exercise2d.pddl`.

Is the number of action landmarks an admissible heuristic for this task? Is the number of fact landmarks (without those already true in the initial state) an admissible estimate? Under which conditions are these statements true in general?

Exercise E.3 (4+2+3+1+2+1 marks)

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

- Variables $V = \{a, b, c, d\}$ with $dom(a) = dom(b) = \{1, 2\}$, $dom(c) = \{1, 2, 3\}$, and $dom(d) = \{1, 2, 3, 4\}$,
- Initial state $I = \{a \mapsto 1, b \mapsto 1, c \mapsto 1, d \mapsto 1\}$,
- Operators $O = \{o_1, \ldots, o_6\}$ where
  - $o_1 = \langle (a = 2) \wedge (b = 1) \wedge (c = 1), (a := 1) \wedge (b := 2) \wedge (c := 3) \rangle$, $cost(o_1) = 3$
  - $o_2 = \langle (a = 1), (a := 2) \rangle$, $cost(o_2) = 5$
  - $o_3 = \langle (b = 1) \wedge (d = 1), (b := 2) \wedge (d := 2) \rangle$, $cost(o_3) = 3$
  - $o_4 = \langle (c = 1) \wedge (d = 1), (c := 2) \wedge (d := 3) \rangle$, $cost(o_4) = 3$
  - $o_5 = \langle (c = 2) \wedge (d = 3), (c := 1) \wedge (d := 4) \rangle$, $cost(o_5) = 1$
  - $o_6 = \langle (c = 2) \wedge (d = 2), (c := 3) \wedge (d := 4) \rangle$, $cost(o_6) = 1$
- Goal $\gamma = (b = 2) \wedge (c = 3) \wedge (d = 4)$.

(a) Use SoPlex to compute an optimal non-negative cost partitioning of the four projections to a, b, c, and d. Please submit the result in three forms: (i) a file encoding the LP with clear names for variables and constraints, (ii) a table with one column per operator and one row per abstraction where each cell contains the cost of the operator in the abstraction, and (iii) the abstract transition systems where edges are annotated with the local cost function and states are annotated with their abstract goal distances. Do not include transitions caused by operators that have no effect on the abstraction.

Instructions on how to install and use SoPlex are in the file `soplex-readme.txt`.

To reduce the work load of this exercise, we provided LaTeX source code for drawing the abstract transition systems in the directory `cost-partitioning` which you can adapt for your solution. This directory also contains a Python file encoding the task, in case you want to create the LP file with a program. However, programming is not required for this exercise and you should not submit source code.

(b) Repeat exercise (a) for a general cost partitioning.

(c) Repeat exercise (a) for computing the posthoc optimization heuristic. The table and graph annotation in this case should show the cost partitioning that is implicitly computed by the heuristic.

(d) Discuss the following question: In exercises (a)–(c) the projection Π^{π_a} always has a heuristic value of 0. Does this mean that it is safe to ignore this abstraction in the cost partitioning?

(e) Discuss the differences between the three ways of partitioning the costs.

(f) Find an unsolvable task where all projections to single variables are solvable but optimal general cost partitioning over the projections shows unsolvability. Explain your example.

Exercise E.4 (2+3+3 marks)

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

- Variables $V = \{a, b, c\}$ with $dom(a) = dom(c) = \{1, 2, 3, 4, 5\}$, and $dom(b) = \{1, 2, 3, 4, 5, 6, 7\}$,
- Initial state $I = \{a \mapsto 1, b \mapsto 1, c \mapsto 1\}$,
- Operators $O = \{o_1, \dots, o_{12}\}$ where
  - $o_1 = \langle (a = 1), (a := 2) \rangle$
  - $o_2 = \langle (a = 2) \land (b = 1) \land (c = 1), (a := 1) \land (b := 3) \rangle$
  - $o_3 = \langle (a = 1) \land (b = 4) \land (c = 1), (b := 2) \land (c := 2) \rangle$
  - $o_4 = \langle (a = 1) \land (b = 3) \land (c = 1), (b := 4) \rangle$
  - $o_5 = \langle (a = 2) \land (b = 2) \land (c = 2), (b := 4) \rangle$
  - $o_6 = \langle (a = 2) \land (b = 4) \land (c = 2), (b := 3) \rangle$
  - $o_7 = \langle (a = 2) \land (b = 3) \land (c = 2), (b := 6) \land (c := 4) \rangle$
  - $o_8 = \langle (a = 2) \land (b = 3) \land (c = 2), (a := 3) \land (b := 5) \rangle$
  - $o_9 = \langle (a = 2) \land (b = 6) \land (c = 2), (a := 4) \land (c := 3) \rangle$
  - $o_{10} = \langle (c = 4), (c := 2) \rangle$
  - $o_{11} = \langle (a = 3) \land (b = 5) \land (c = 4), (a := 5) \land (b := 7) \land (c := 5) \rangle$
  - $o_{12} = \langle (a = 4) \land (b = 6) \land (c = 3), (a := 5) \land (b := 7) \land (c := 5) \rangle$

  and $cost(o) = 1$ for all $o \in O$,
- Goal $\gamma = (a = 5) \land (b = 7) \land (c = 5)$.

(a) Provide the LP solved by the flow heuristic for I as an input file for SoPlex (see exercise E.3). Use each atom as the constraint name for its flow constraint so it is easy to see which constraint belongs to which atom. Then solve the LP and provide the objective value.

As for exercise E.3, Python code encoding the task is available in the directory `network-flow` *but no programming is required.*

(b) Draw the transition systems of the three projections to a, b, and c. For operators that do not mention the variable, include just one representative self-loop at the goal state to keep the transition system concise. Annotate each edge with the flow of the operator from exercise (a) and highlight edges with a non-zero value. What do you notice in the abstractions? Discuss your observations.

As for exercise E.3, LaTeX source code for drawing the abstract transition systems is available in the directory `network-flow`*.*

(c) Let $\Pi = \langle V, I, O, \gamma \rangle$ be a task in TNF with atoms A. Let h^{flow} be the flow heuristic and h^{pot} an admissible and consistent potential heuristic with atomic features that maximizes the heuristic value of I. The general form of the LP computed for $h^{\text{flow}}(I)$ can be written like this:

$$\text{Minimize } \sum_{o \in O} \mathit{cost}(o)\mathsf{Count}_o \text{ subject to}$$

$$\sum_{o \in O}([a \in \mathit{eff}(o)] - [a \in \mathit{pre}(o)])\mathsf{Count}_o = [a \in \gamma] - [a \in I] \quad \text{for all } a \in A$$

$$\mathsf{Count}_o \geq 0 \quad \text{for all } o \in O.$$

Compare the dual of this LP to the LP computed for $h^{\text{pot}}(I)$.

Exercise E.5 (7 marks)

In this exercise, we want to draw a connection from linear programming to delete-relaxed planning tasks. In particular, the paper below describes an IP model for the delete relaxation. Explain the IP in detail, i.e., explain the meaning of all involved variables and constraints. Use your own words and the notation from the lecture instead of the notation from the paper. Also explain how the IP can be extended with operator counting constraints, again using notation from the lecture.

Tatsuya Imai and Alex Fukunaga. A Practical, Integer-Linear Programming Model for the Delete-Relaxation in Cost-Optimal Planning. In *Proc. ECAI 2014*. pp. 459–464, 2014.

A good answer can be written in about 1 page.

Exercise E.6 (5 marks)

Describe the following heuristics as potential heuristics by defining appropriate state features and a weight for each feature. When defining the features, make reasonable assumptions about the encoding of the problem and describe them as well.

- Use atomic features to encode the Manhattan distance heuristic in the sliding tile puzzle.
- Use binary features to encode the "last move enhancement" of the Manhattan distance heuristic in the sliding tile puzzle. (We do not consider linear conflicts here, so ignore issues arising from a tile influencing both the last move and a linear conflict.) A binary feature $f_{X=x \wedge Y=y}$ is defined analogously to a atomic feature as

$$f_{X=x \wedge Y=y}(s) = \begin{cases} 1 & \text{if s[X] = x and s[Y] = y} \\ 0 & \text{otherwise.} \end{cases}$$

- Use binary features to encode the "corner enhancement" of the Manhattan distance heuristic in the sliding tile puzzle.
- Use atomic features to encode the goal-counting heuristic in an SAS$^+$ task.
- Use atomic features to encode the material value of a chess position.

You can find details about the sliding tile puzzle, the Manhattan distance heuristics and its enhancements in the following paper:

Richard Korf and Larry Taylor. Finding optimal solutions to the twenty-four puzzle. In *Proc. AAAI 1996*, pp. 1202–1207, 1996.

For information on the material value of a chess position see for example: `https://en.wikipedia.org/wiki/Chess_piece_relative_value`