Planning and Optimization

G. Röger, T. Keller
G. Francès

University of Basel
Fall Semester 2018

Exercise Sheet E

Due: November 25, 2018

The files required for this exercise are in the directory `exercise-e` *of the course repository (*`https://bitbucket.org/aibasel/planopt-hs18`*). All paths are relative to this directory. Update your clone of the repository with* `hg pull -u` *to see the files.*

Exercise E.1 (5 marks)

In the lecture we have discussed the use of disjunctive action landmarks to compute admissible heuristics such as the h^{MHS} or the $h^{\text{LM-cut}}$ heuristic, the second of which is a state-of-the-art heuristic for optimal planning. Historically, however, other uses of landmarks in the context of satisficing planning predate the development of the above heuristics. In this exercise, we ask you to read a conference paper describing one such use, and to discuss it:

> Silvia Richter, Malte Helmert and Matthias Westphal. Landmarks Revisited. In *Proc. AAAI*, pp. 975-982, 2008.

Read the paper and summarize in your own words (between 300 and 600 words) its main contributions. Your summary should briefly look at at least the following key issues:

- What kind of landmarks does the previous work by Hoffmann, Porteous and Sebastia (2003) use? How do they differ from the disjunctive action landmarks we have seen in class?
- How are landmarks generated in LM^{RPG}? How does exactly LM^{RPG} make sure that a given problem fact is a landmark?
- How are landmarks used in LM^{RPG}?
- How does the contribution by Richter, Helmert and Westphal compare to the three issues above? How do they generate the landmarks? What kind of landmarks do they use? How do they use them? Why do the call their contribution a "pseudo-heuristic"?
- What conclusions can you draw from the empirical results?

Exercise E.2 (4+3 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`. What is the heuristic value of the initial state in a task such as `benchmarks/blocks/probBLOCKS-9-0.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 the task in the directory `lmcut`. 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.

Exercise E.3 (3 marks)

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

- Variables $V = \{a, b, c\}$ with $dom(a) = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $dom(b) = \{1, 2, 3, 4, 5, 6, 7\}$, and $dom(c) = \{1, 2, 3, 4, 5\}$,
- Initial state $I = \{a \mapsto 1, b \mapsto 1, c \mapsto 1\}$,
- Operators $O = \{o_1, \ldots, o_{14}\}$ where
  - $o_1 = \langle (a = 1) \wedge (b = 1) \wedge (c = 1), (a := 2) \wedge (b := 2) \wedge (c := 2) \rangle$
  - $o_2 = \langle (a = 2) \wedge (b = 2) \wedge (c = 2), (a := 2) \wedge (b := 4) \wedge (c := 3) \rangle$
  - $o_3 = \langle (a = 2) \wedge (b = 2) \wedge (c = 2), (a := 3) \wedge (b := 3) \wedge (c := 2) \rangle$
  - $o_4 = \langle (a = 2) \wedge (b = 3) \wedge (c = 2), (a := 4) \wedge (b := 3) \wedge (c := 4) \rangle$
  - $o_5 = \langle (a = 3), (a := 2) \rangle$
  - $o_6 = \langle (b = 3), (b := 4) \rangle$
  - $o_7 = \langle (a = 3) \wedge (b = 3) \wedge (c = 3), (a := 5) \wedge (b := 5) \wedge (c := 5) \rangle$
  - $o_8 = \langle (a = 4) \wedge (b = 4) \wedge (c = 4), (a := 5) \wedge (b := 5) \wedge (c := 5) \rangle$
  - $o_9 = \langle (a = 5), (a := 6) \rangle$
  - $o_{10} = \langle (a = 6) \wedge (b = 5), (a := 5) \wedge (b := 6) \rangle$
  - $o_{11} = \langle (a = 6) \wedge (b = 6), (a := 7) \wedge (b := 6) \rangle$
  - $o_{12} = \langle (a = 7) \wedge (b = 7), (a := 6) \wedge (b := 7) \rangle$
  - $o_{13} = \langle (a = 7) \wedge (b = 6), (a := 8) \wedge (b := 7) \rangle$
  - $o_{14} = \langle (a = 8) \wedge (b = 7), (a := 7) \wedge (b := 7) \rangle$

  and $cost(o) = 1$ for all $o \in O$,
- Goal $\gamma = (a = 6) \wedge (b = 7) \wedge (c = 5)$.

Provide the LP solved by the flow heuristic for I as an input file for the solver *lp-solve*. 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 and values for all variables Count_o in the discovered solution.
On Ubuntu you can install lp-solve with `sudo apt install lp-solve`. *Its input format is described on* `http://lpsolve.sourceforge.net/5.5/lp-format.htm` *and in the example file in the directory* `lp`. *You can run lp-solve using* `lp_solve /path/to/file.lp`.

Exercise E.4 (4+3 marks)

(a) 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 atomic 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.)
- 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`

(b) Consider the linear constraints that characterize admissible and consistent atomic potential heuristics. Find a parametrized objective function for each of the following use cases. To be suitable as an objective function in an LP solver, the function should be linear in the weights and the number of its coefficients should be polynomial in the number of atoms (and the number of sample states where we use them).

- Maximize the heuristic value of the initial state.
- Maximize the sum of heuristic values for the states in a given set of sample states S.
- Maximize the average of heuristic values of all states (including unreachable states).

Exercise E.5 (4+2 marks)

(a) Consider the task from exercise E.3 and the three projections to a, b, and c. Compute an optimal cost partitioning and an optimal general cost partitioning for the abstractions. In each case, provide the cost partitioning as a table and the abstract transition systems of the projections. In the projections, annotate edges with their cost and states with their goal distances under the cost partitioning.

Discuss the differences between the two ways of partitioning the costs.

(b) In the lecture, we have seen how to compute a uniform cost partitioning for a set of disjunctive action landmarks (chapter E1). Analogously, the set of disjunctive action landmarks which is built during the computation of the LM-cut heuristic induces a (non-uniform) cost partitioning. For a sequence $\langle L_1, \ldots, L_n \rangle$ of disjunctive landmarks that is computed by LM-cut, formalize the induced cost partitioning $\langle cost_{L_1}, \ldots, cost_{L_n} \rangle$ by LM-cut.

The exercise sheets can be submitted in groups of three students. Please submit one single copy of the exercises per group (only one member of the group does the submission), and provide all student names on the submission.