Planning and Optimization

G. Röger, T. Keller
G. Francès

University of Basel
Fall Semester 2018

Exercise Sheet B

Due: October 21, 2018

The files required for this exercise are in the directory `exercise-b` *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 B.1 (3 marks)

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

$$
\begin{aligned}
V &= \{a, b, c, d\} \\
I(v) &= \mathbf{F} \quad \text{for all } v \in V \\
O &= \{o_1, o_2, o_3, o_4\} \\
\gamma &= d
\end{aligned}
$$

and

$$
\begin{aligned}
o_1 &= \langle \neg b, c \rangle \\
o_2 &= \langle \top, b \rangle \\
o_3 &= \langle \neg a, a \rangle \\
o_4 &= \langle a, (b \rhd d) \wedge \neg c \rangle
\end{aligned}
$$

Plot the search space explored by a progression and by a regression breadth-first search through this task. In the regression search simplify the state formula as much as possible at every node of the search tree. Do not expand the node further if that formula is unsatisfiable or logically entails the state formula of a previously encountered node. In the progression search do not expand a node if its state is a duplicate of a previously encountered state.

Exercise B.2 (5+2+3+2+2 marks)

(a) In the file `regression/strips_regression.py` you will find a partial implementation of a breadth-first regression search for STRIPS tasks. Complete the missing parts and use it to solve the Beleaguered Castle instance from the last exercise (a grounded PDDL model in STRIPS of it is in the directory `castle`). Only regress a formula through an operator if that operator adds at least one proposition of the formula. Ignore search states with an unsatisfiable formula or a formula that is equivalent to the formula of a previously encountered state. You don't have to ignore formulas that imply the formula of a previously encountered state but are not equivalent (i.e., represent strict subsets of states).

How many states are generated and expanded? Have a look at some of the generated states and explain why so many states are expanded.

(b) Extend your code from exercise (a) with mutex-based pruning by completing the following steps:

- Complete the method `create_mutexes` by mapping each proposition to a set of propositions that are mutually exclusive with it. Normally, such mutex groups would be discovered automatically from the planning task but here you can manually add mutex groups for the specific instance. Use the mutex groups from exercise A.1 (d).
- Before starting the search, call `create_mutexes` and store the result.

- Before inserting a node into the queue, loop over all propositions in the formula. For each proposition check if the set of propositions mutex with it intersects the formula. If it does, there are two mutually exclusive propositions in the formula and it does not have to be added to the queue.

Repeat the experiment from exercise (a) and discuss the differences.

(c) The file `regression/general_regression.py` contains a partial implementation of general regression of a formula through an effect. Complete the implementation and use it on the task `vampire/p01_grounded.pddl`. Regress the goal through each operator and list the resulting formulas. Simplify the formulas as much as possible (within your implementation or manually). If a formula violates mutexes, list at least one of the violated mutexes (without proof).

(d) Provide a family of planning tasks Π_n such that the size of Π_n is polynomial in n, and such that a breadth-first search with regression expands only a polynomial number of search nodes in n, whereas a breadth-first search with progression needs to expand an exponential number of search nodes in n. Assume the progression search prunes all duplicate states and the regression prunes a state if its formula logically entails the formula of its parent.

(e) Provide a family of planning tasks Π_n such that the size of Π_n is polynomial in n, and such that a breadth-first search with progression expands only a polynomial number of search nodes in n, whereas a breadth-first search with regression needs to expand an exponential number of search nodes in n. Assume the same pruning as in exercise (d).

Exercise B.3 (2+3 marks)

(a) Prove BCPlanEx $\leq_{\mathrm{p}}$ PlanEx. You may use the results from the lecture.

(b) Your friend tells you their new idea of how to test whether a planning task is solvable:

> "So, you start with parsing the PDDL file and throw away all of the costs; they are not important. Also, you have to change the task so it only has one goal state. To do this you just [...] *[your friend's description of how to do this is a little hard to follow]*. Then you create a graph with cities that represent [...] *[again this is hard to understand]*. You then have to compute a number K and distances for all the pairs of cities by [...] *[you start to think your friend is not the best at explaining this idea]*. Then compute the cost of the shortest tour through all of these cities; there are super-fast solvers for that, like Concorde. If the tour is cheaper than K then the task is solvable otherwise it is unsolvable."

Discuss this idea (you do *not* have to come up with a way to fill the gaps). Is it possible that such a method can work? Are the simplifications at the start justified? Assuming $\mathsf{P} \neq \mathsf{NP} \neq \mathsf{PSPACE}$ what can you say about the time to create the graph?

The exercise sheets can be submitted in groups of three students. Please provide all student names on the submission.