

Planning and Optimization

M. Helmert, G. Röger
T. Keller, M. Wehrle

University of Basel
Fall Semester 2016

Classroom Exercise 2

For the runs with Fast Downward, set a time limit of 1 minute and a memory limit of 2 GB. Using Linux, such limits can be set with `ulimit -t 60` and `ulimit -v 2000000`, respectively.

Exercise 1

The goal of this exercise is to implement the A* search algorithm in the Fast Downward planner. We have prepared a stub of the class `AStarSearch`, which can be added to your Fast Downward repository directly via mercurial. To do so, change to the directory where you have installed Fast Downward and execute the commands

```
$ hg pull https://tkeller@bitbucket.org/aibasel/planopt-hs16
$ hg update planopt-hs16
```

Implement the parts marked as missing with a comment `“// insert your code here”`:

- Add a comparison operator for two objects of type `AStarSearchNode`. You can find the stub in `src/search/search_engines/astar_search.h` in the function

```
bool operator()(const AStarSearchNode *lhs, const AStarSearchNode *rhs) const
```

of the struct `Compare` within the `AStarSearchNode` class. Implement the comparison such that it returns `true` if the f-value of `lhs` is larger than the f-value of `rhs`, or if both f-values are equal and the h-value of `lhs` is larger than the h-value of `rhs`.

- The class `OpenList` internally works with the priority queue `std::priority_queue` from the C++ standard library. Based on the corresponding functions in `std::priority_queue`, implement the two functions to insert and remove `AStarSearchNodes` into and from `OpenList` in `src/search/search_engines/astar_search.h`.
- In `src/search/search_engines/astar_search.cc`, the implementation of the A* search algorithm in the `search` method of `AStarSearch` is missing. Implement A* based on the pseudo code that has been presented in the lecture.

Your implementation of A* based on the provided code stub is connected to a fixed admissible heuristic (the *LM-Cut* heuristic), which is invoked automatically. Test the correctness of your implementation (called with `--search "planopt_astar()"`) by comparing to the Fast Downward version of A* and LM-Cut (called with `--search "astar(lmcut())"`) on all instances in the benchmarks directory. Discuss the search time and the plan costs.

Please form groups of two students for classroom exercises.