River-crossing Problems and Planning

Bob de Ruiter / s4344952
SPML Assignment 2

Background

One of the most famous river-crossing problems dates from the first millenium:

> *A man has to take his wolf, goat, and cabbage across a river. His boat can only carry him and one of his belongings. If the cabbage and the goat are left together unattended, the goat will eat the cabbage. If the wolf and the goat are left unattended, the wolf will eat the goat.*
>
> *How can the man get everything to the other side of the river?*

The general make-up of all classic river-crossing problems is the same. In the initial state, a group of people and things is at one side of the river, along with a single boat. The goal state is to have all people and things at the other side. At least one person must be on the boat to move it across the river. The boat can only support a limited number of passengers and belongings.

The difference between river-crossing problems lies in the composition of the group and the conditions under which certain people and things can stay together on either side of the river or on the boat.

I originally set out to capture most classic river-crossing problems in a single STRIPS-compatible PDDL domain, but this turned out to be incredibly difficult, if not downright impossible. Quantifiers and negative preconditions, both of which are missing in STRIPS, are required to encompass the entire problem space. I also looked at the Action Description Language, an extension of PDDL which supports both negation and quantifiers, but I was unable to find a library of heuristics for ADL similar to those included in pyperplan. In the end, I only implemented the wolf/goat/cabbage problem. This was not trivial since although quantifiers are no longer necessary, negation is still present: the cabbage and goat cannot be at the same side, for example.

I hypothesized that since the problem is quite constrained (the farmer often has only one option), the different search algorithms and heuristics may have very similar performance (measured in nodes expanded).

Methods

I wrote a Bash script to try all problems:

```
#!/bin/bash

for h in {lmcut,blind,landmark,hadd,hff,hmax,hsa}; do
    for s in {astar,wastar,gbf,bfs,ehs,ids}; do
        echo $h
        echo $s
        ./src/pyperplan.py -H $h -s $s domain.pddl problem.pddl |
grep expanded
    done
done
```

Results

For each heuristic, greedy best first search is (one of the) the best-performing search algorithms. For each search algorithm, hadd is the best-performing heuristic. As expected, the difference between the best and worst performance is retively small (24 vs 31 nodes expanded). The full results can be found in Appendix A.

Conclusion

Although the choice of algorithms and heuristics makes a large difference in many search problems, this was not the case here.

Appendix A

```
lmcut
astar
2017-04-22 20:55:22,525 INFO     30 Nodes expanded
lmcut
wastar
2017-04-22 20:55:22,767 INFO     30 Nodes expanded
lmcut
gbf
2017-04-22 20:55:23,006 INFO     30 Nodes expanded
lmcut
bfs
2017-04-22 20:55:23,178 INFO     30 Nodes expanded
lmcut
```

```
ehs
2017-04-22 20:55:23,401 INFO     30 Nodes expanded
lmcut
ids
2017-04-22 20:55:23,578 INFO     31 Nodes expanded
blind
astar
2017-04-22 20:55:23,748 INFO     30 Nodes expanded
blind
wastar
2017-04-22 20:55:23,916 INFO     30 Nodes expanded
blind
gbf
2017-04-22 20:55:24,085 INFO     30 Nodes expanded
blind
bfs
2017-04-22 20:55:24,257 INFO     30 Nodes expanded
blind
ehs
2017-04-22 20:55:24,452 INFO     30 Nodes expanded
blind
ids
2017-04-22 20:55:24,628 INFO     31 Nodes expanded
landmark
astar
2017-04-22 20:55:24,813 INFO     30 Nodes expanded
landmark
wastar
2017-04-22 20:55:24,986 INFO     30 Nodes expanded
landmark
gbf
2017-04-22 20:55:25,161 INFO     27 Nodes expanded
landmark
bfs
2017-04-22 20:55:25,328 INFO     30 Nodes expanded
landmark
ehs
2017-04-22 20:55:25,511 INFO     30 Nodes expanded
landmark
ids
2017-04-22 20:55:25,685 INFO     31 Nodes expanded
hadd
astar
```

```
2017-04-22 20:55:25,863 INFO     24 Nodes expanded
hadd
wastar
2017-04-22 20:55:26,045 INFO     24 Nodes expanded
hadd
gbf
2017-04-22 20:55:26,226 INFO     24 Nodes expanded
hadd
bfs
2017-04-22 20:55:26,404 INFO     30 Nodes expanded
hadd
ehs
2017-04-22 20:55:26,586 INFO     29 Nodes expanded
hadd
ids
2017-04-22 20:55:26,762 INFO     31 Nodes expanded
hff
astar
2017-04-22 20:55:26,944 INFO     30 Nodes expanded
hff
wastar
2017-04-22 20:55:27,130 INFO     30 Nodes expanded
hff
gbf
2017-04-22 20:55:27,311 INFO     24 Nodes expanded
hff
bfs
2017-04-22 20:55:27,498 INFO     30 Nodes expanded
hff
ehs
2017-04-22 20:55:27,677 INFO     30 Nodes expanded
hff
ids
2017-04-22 20:55:27,852 INFO     31 Nodes expanded
hmax
astar
2017-04-22 20:55:28,034 INFO     30 Nodes expanded
hmax
wastar
2017-04-22 20:55:28,218 INFO     30 Nodes expanded
hmax
gbf
2017-04-22 20:55:28,402 INFO     30 Nodes expanded
```

```
hmax
bfs
2017-04-22 20:55:28,578 INFO     30 Nodes expanded
hmax
ehs
2017-04-22 20:55:28,758 INFO     30 Nodes expanded
hmax
ids
2017-04-22 20:55:28,932 INFO     31 Nodes expanded
hsa
astar
2017-04-22 20:55:29,115 INFO     30 Nodes expanded
hsa
wastar
2017-04-22 20:55:29,303 INFO     30 Nodes expanded
hsa
gbf
2017-04-22 20:55:29,507 INFO     30 Nodes expanded
hsa
bfs
2017-04-22 20:55:29,677 INFO     30 Nodes expanded
hsa
ehs
2017-04-22 20:55:29,859 INFO     30 Nodes expanded
hsa
ids
2017-04-22 20:55:30,034 INFO     31 Nodes expanded
```