Classical Planning Algorithms

4. Peeking Inside: Implementing a Simple Heuristic

Malte Helmert

ICAPS 2018 Summer School

June 21, 2018

Implementing a Heuristic

Hamming Distance Heuristic

- heuristic $h : S \to \mathbb{R}_0^+$
  estimates cost to reach a goal state from a given state
- easy to extend Fast Downward with new heuristics
- simple example: Hamming distance heuristic counts number of variables that do not have required goal value

Example:

- $s = \{A \mapsto a, B \mapsto b, C \mapsto c\}$
- $s_\star = \{A \mapsto a, B \mapsto b'\}$

$\rightsquigarrow$ $h^{\text{Hamming}}(s) = 1$

Steps to Add a Heuristic

Adding a Heuristic to Fast Downward

relative to directory `/vagrant/fast-downward`:

1. Create a new `*.cc` file in `src/search/heuristics` that implements the heuristic.
2. Edit `src/search/DownwardFiles.cmake` to add a CMake plug-in definition for the new source file.
3. Rebuild the planner.

You might prefer to work on a copy of `/vagrant/fast-downward` to keep the original working planner around.

Step 1: Add the C++ Implementation of the Heuristic

Demo

```
$ cd /vagrant/fast-downward/src/search/heuristics
$ emacs hamming_heuristic.cc
$ # type, type, type! (file does not yet exist)
```

- header file (`*.h`) optional
- derive a new subclass from class `Heuristic`
- implement constructor and `compute_heuristic` method
- create a static plug-in object to make the heuristic available on the command line

Note: finished implementation provided at
`/vagrant/lectures/classical/code/hamming_heuristic.cc`

Step 2: Tell CMake About the New Source File

CMake plug-in definition:

CMake Module for Hamming Heuristic

```
fast_downward_plugin(
    NAME HAMMING_HEURISTIC
    HELP "The Hamming distance heuristic"
    SOURCES
        heuristics/hamming_heuristic
    DEPENDS TASK_PROPERTIES
    )
```

- `HELP` and `DEPENDS` lines can be omitted
  $\rightsquigarrow$ only needed for advanced CMake features (custom builds)
- `NAME` only used internally within CMake

Step 2: Tell CMake About the New Source File

Demo

```
$ cd /vagrant/fast-downward/src/search
$ emacs DownwardFiles.cmake &
```

Note: modified CMake file provided at
`/vagrant/lectures/classical/code/DownwardFiles.cmake`

Step 3: Rebuild the Planner

Demo

```
$ cd /vagrant/fast-downward
$ ## force full rebuild (only do this if necessary!):
$ # rm -rf builds
$ ## build the planner in release64 configuration:
$ ./build.py release64
...
```

- only force full rebuild after changes to CMake files
- even with CMake changes full rebuild not always needed
- full rebuilds take a long time

Testing the New Heuristic

Demo

```
$ cd /vagrant/lectures/classical/demo
$ ./fd tile/cheat.pddl tile/cheat01.pddl \
      --search "eager_greedy([hamming()])"
...
Hamming heuristic says hello!
...
Solution found!
...
```

Reminder: The `./fd` shell script is an alias for
`/vagrant/fast-downward/fast-downward.py --build=release64`

Summary and Outlook

Summary and Outlook

That's it from me!

We talked about:

- What is Classical Planning?
- Planning Formalisms and Models
- Planning Algorithms
- Peeking Inside: Implementing a Simple Heuristic

tomorrow:

- 9:00–10:30: Lecture Classical Planning Heuristics
- 16:00–open: Lab Classical and Probabilistic Planning

The End

Thank you for your attention!