Planning and Optimization

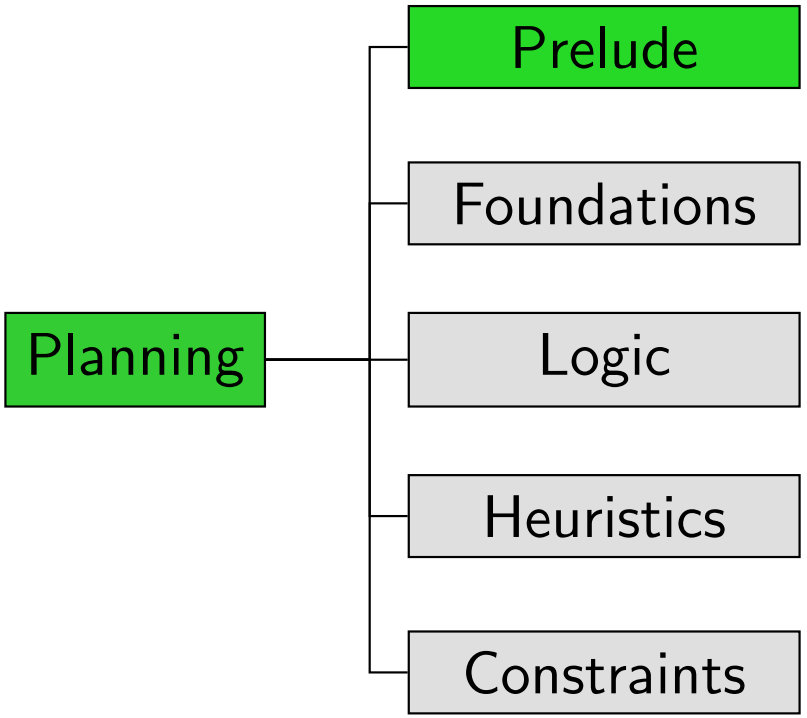
A3. Getting to Know a Planner

Malte Helmert and Gabriele Röger

Universität Basel

September 26, 2022

Content of this Course



Fast Downward, VAL and INVAL

Getting to Know a Planner

We now play around a bit with a planner and its input:

- look at problem formulation
- run a planner (= planning system/planning algorithm)
- validate plans found by the planner

Planner: Fast Downward

Fast Downward

We use the Fast Downward planner in this course

- because we know it well (developed by our research group)
- because it implements many search algorithms and heuristics
- because it is the classical planner most commonly used as a basis for other planners

$\rightsquigarrow$ `https://www.fast-downward.org`

Validator: VAL

VAL

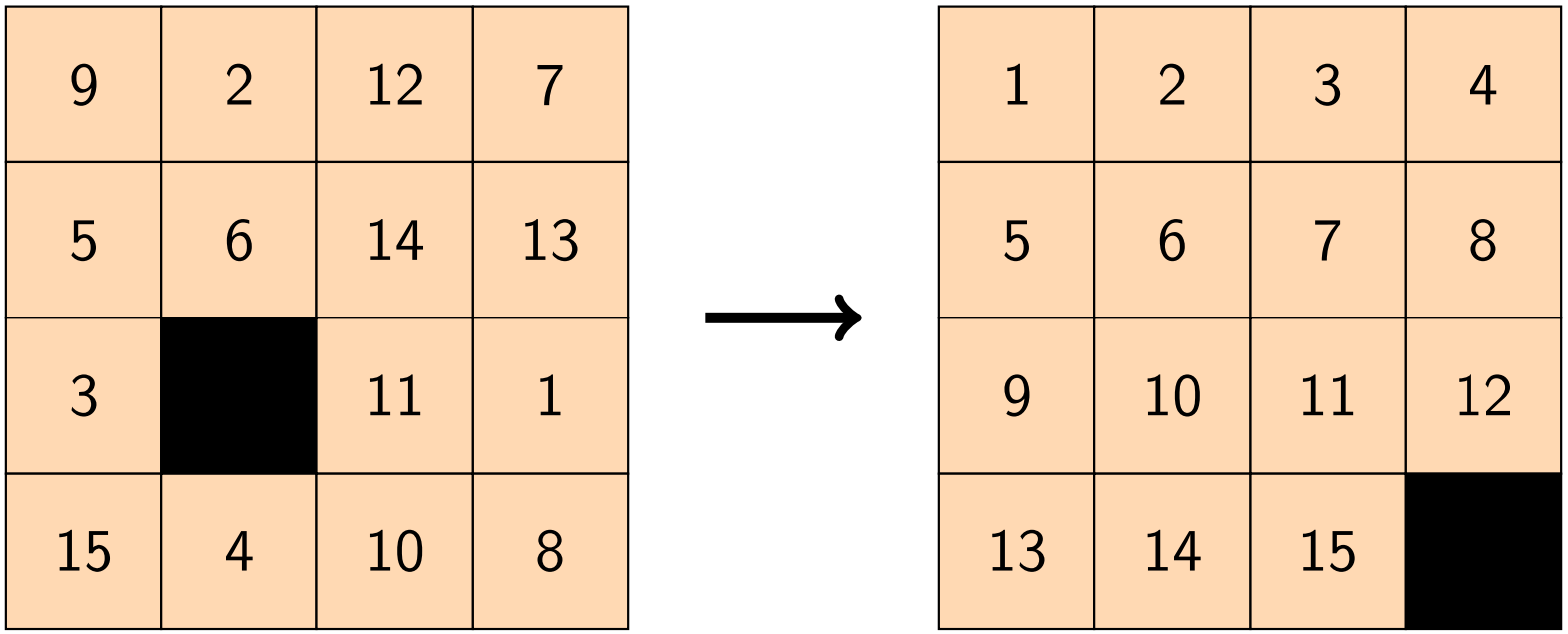
We use the VAL plan validation tool (Fox, Howey & Long) to independently verify that the plans we generate are correct.

- very useful debugging tool
- `https://github.com/KCL-Planning/VAL`

Because of bugs/limitations of VAL, we will also occasionally use another validator called INVAL (by Patrik Haslum).

15-Puzzle

Illustrating Example: 15-Puzzle



Solving the 15-Puzzle

Demo

```
$ cd demo
$ less tile/puzzle.pddl
$ less tile/puzzle01.pddl
$ ./fast-downward.py \
      tile/puzzle.pddl tile/puzzle01.pddl \
      --heuristic "h=ff()" \
      --search "eager_greedy([h],preferred=[h])"
...
$ validate tile/puzzle.pddl tile/puzzle01.pddl \
      sas_plan
...
```

Variation: Weighted 15-Puzzle

Weighted 15-Puzzle:

- moving different tiles has different cost
- cost of moving tile x = number of prime factors of x

Demo

```
$ cd demo
$ meld tile/puzzle.pddl tile/weight.pddl
$ meld tile/puzzle01.pddl tile/weight01.pddl
$ ./fast-downward.py \
      tile/weight.pddl tile/weight01.pddl \
      --heuristic "h=ff()" \
      --search "eager_greedy([h],preferred=[h])"
...
```

Variation: Glued 15-Puzzle

Glued 15-Puzzle:

- some tiles are glued in place and cannot be moved

Demo

```
$ cd demo
$ meld tile/puzzle.pddl tile/glued.pddl
$ meld tile/puzzle01.pddl tile/glued01.pddl
$ ./fast-downward.py \
      tile/glued.pddl tile/glued01.pddl \
      --heuristic "h=cg()" \
      --search "eager_greedy([h],preferred=[h])"
...
```

Note: different heuristic used!

Variation: Cheating 15-Puzzle

Cheating 15-Puzzle:

- Can remove tiles from puzzle frame (creating more blanks) and reinsert tiles at any blank location.

Demo

```
$ cd demo
$ meld tile/puzzle.pddl tile/cheat.pddl
$ meld tile/puzzle01.pddl tile/cheat01.pddl
$ ./fast-downward.py \
      tile/cheat.pddl tile/cheat01.pddl \
      --heuristic "h=ff()" \
      --search "eager_greedy([h],preferred=[h])"
...
```

Summary

Summary

- We saw planning tasks modeled in the PDDL language.
- We ran the Fast Downward planner and VAL plan validator.
- We made some modifications to PDDL problem formulations and checked the impact on the planner.