Planning and Reasoning
Final project presentation

Andrea Gervasio

SAPIENZA
UNIVERSITÀ DI ROMA

Domain definition

- A robot is placed in an environment made of different rooms
  - The rooms can be locked
  - The robot can’t enter in a locked room
  - Each room can be unlocked using a specific key

- The goal is to clean all the rooms
  - Each room can have dust or stains on the floor (or both)
  - The dust can be cleaned with a sweeper
  - The stains on the floor can be cleaned with a mop
  - The robot needs the specific tool to clean the filth
  - The rooms can be more or less dirty

Domain implementation

Planner: Fast-Downward (no *:fluents*)

- **Requirements:**
  - ***:strips***
  - ***:typing***: used to define types to initialize objects and function parameters
  - ***:action-costs***: used to define a *total-cost to minimize*

- **Types:**
  - *Room*
  - *Robot*
  - *Quantity*
  - *Key*
  - *Sweeper*
  - *Mop*

```
(define (domain cleaning_rooms_with_keys_and_pickup)
  (:requirements :strips :typing :action-costs)

  (:types
   room - object
   robot - object
   quantity - object
   key - object
   sweeper - object
   mop - object
   )
```

Domain implementation

- **Predicates and functions:**

```
(:predicates
 (at-robot ?r - robot ?loc - room)
 (at-key ?k - key ?loc - room)
 (at-mop ?m - mop ?loc - room)
 (at-sweeper ?s - sweeper ?loc - room)
 (has-key ?r - robot ?k - key)
 (has-dust ?room - room)
 (has-stains ?room - room)
 (has-sweeper ?r - robot ?s - sweeper)
 (has-mop ?r - robot ?m - mop)
 (dirty ?room - room)
 (to-be-cleaned ?room - room ?q - quantity)
 (locked ?room - room)
 (key-opens ?k - key ?lock - room)
 (plus1 ?q1 ?q2 - quantity)
 )

(:functions
 (distance ?l1 ?l2 - room)
 (total-cost)
 (cost-of ?q - quantity)
 )
```

Domain implementation

- **Actions:**

```
(:action pick-up-key
 :parameters (?r - robot ?room - room ?key - key)
 :precondition (and (at-robot ?r ?room) (at-key ?key ?room))
 :effect (and (not (at-key ?key ?room)) (has-key ?r ?key)
       (increase (total-cost) 1))
 )

(:action pick-up-mop
 :parameters (?r - robot ?room - room ?mop - mop)
 :precondition (and (at-robot ?r ?room) (at-mop ?mop ?room))
 :effect (and (not (at-mop ?mop ?room)) (has-mop ?r ?mop)
       (increase (total-cost) 1))
 )

(:action pick-up-sweeper
 :parameters (?r - robot ?room - room ?sweeper - sweeper)
 :precondition (and (at-robot ?r ?room) (at-sweeper ?sweeper ?room))
 :effect (and (not (at-sweeper ?sweeper ?room)) (has-sweeper ?r ?sweeper)
       (increase (total-cost) 1))
 )
```

Domain implementation

- **Actions:**

```
(:action move
 :parameters (?r - robot ?from ?to - room)
 :precondition (and (at-robot ?r ?from) (not (locked ?to)))
 :effect (and (not (at-robot ?r ?from))
              (at-robot ?r ?to)
              (increase (total-cost) (distance ?from ?to)))
 )

(:action unlock
 :parameters (?r - robot ?room - room ?k - key)
 :precondition (and (locked ?room) (has-key ?r ?k) (key-opens ?k ?room))
 :effect (and (not (locked ?room))
       (increase (total-cost) 1))
 )
```

Domain implementation

- **Actions:**

```
(:action sweep
 :parameters (?r - robot ?room - room ?q - quantity ?s - sweeper)
 :precondition (and (at-robot ?r ?room) (dirty ?room) (has-dust ?room)
                    (to-be-cleaned ?room ?q) (has-sweeper ?r ?s))
 :effect (and (when (not (has-stains ?room))
                (and (not (dirty ?room)) (not (to-be-cleaned ?room ?q))))
              (not (has-dust ?room))
              (increase (total-cost) (cost-of ?q)))
 )

 (:action clean-mop
 :parameters (?r - robot ?room - room ?q - quantity ?m - mop)
 :precondition (and (at-robot ?r ?room) (dirty ?room) (has-stains ?room)
                    (to-be-cleaned ?room ?q) (has-mop ?r ?m))
 :effect (and (when (not (has-dust ?room))
                (and (not (dirty ?room)) (not (to-be-cleaned ?room ?q))))
              (not (has-stains ?room))
              (increase (total-cost) (cost-of ?q)))
 )
```

Heuristics: Blind heuristic

The blind heuristic of the Fast-Downward planner is defined as follows:

For each state *s* in *S*:

$$h(s) = \begin{cases} \min_{a \text{ in } A} \text{cost}(a) & \text{if } s \text{ in not in } S_G \\ 0 & \text{otherwise} \end{cases}$$

The blind heuristic is admissible, consistent and safe.

Heuristics: h^{max}

The h^{max} heuristic is defined as follows:

Computing h^{max} Values

Associate a *cost* attribute with each node.
for all nodes n:
    $n.cost := \infty$
while no fixed point is reached:
    Choose a node n.
    if n is an AND node that is not an effect node:
        $n.cost := \max_{n' \in succ(n)} n'.cost$
    if n is an effect node for operator o:
        $n.cost := cost(o) + \max_{n' \in succ(n)} n'.cost$
    if n is an OR node:
        $n.cost := \min_{n' \in succ(n)} n'.cost$

The h^{max} heuristic is admissible, consistent, safe and goal-aware.

Heuristics: h^{FF}

The h^{FF} heuristic is defined as follows:

Definition (FF Heuristic)

Let $\Pi = \langle V, I, O, \gamma \rangle$ be a propositional planning task in positive normal form. The FF heuristic for a state s of Π, written $h^{\mathrm{FF}}(s)$, is computed as follows:

- Construct the RTG for the task $\langle V, s, O^{+}, \gamma \rangle$
- Construct the best achiever graph G^{add}.
- Compute the set of effect nodes $\{n_{o_1}^{\chi_1}, \ldots, n_{o_k}^{\chi_k}\}$ reachable from n_γ in G^{add}.
- Return $h^{\mathrm{FF}}(s) = \sum_{i=1}^{k} cost(o_i)$.

The h^{FF} heuristic is neither admissible nor consistent, but it is safe and goal-aware.

Problem instances: 1

There are three rooms and one robot in the environment. The robot already has all the keys and all the cleaning tools in the initial state. The robot needs to return to the room where it started (R1) after cleaning all the rooms.

Objects:

```
(:objects
  R1 R2 R3 - room
  CleaningBot - robot
  n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 - quantity
  Key1 Key2 - key
  Mop - mop
  Sweeper - sweeper
)
```

Initial state:

```
(:init
  (at-robot CleaningBot R1)
  (dirty R1)
  (dirty R2)
  (dirty R3)
  (to-be-cleaned R1 n6)
  (to-be-cleaned R2 n3)
  (to-be-cleaned R3 n9)
  (key-opens Key2 R2)
  (key-opens Key1 R3)
  (has-key CleaningBot Key1)
  (has-key CleaningBot Key2)
  (has-mop CleaningBot Mop)
  (has-sweeper CleaningBot Sweeper)
  (has-dust R1)
  (has-dust R2)
  (has-stains R2)
  (has-stains R3)
)
```

Problem instances: constants

There are some values which are the same in all 5 problems. I will report here the most complete version, namely the one of the fifth problem.
If an instance does not have all the elements defined, the corresponding lines are simply erased.

```
(plus1 n1 n2)
(plus1 n2 n3)
(plus1 n3 n4)
(plus1 n4 n5)
(plus1 n5 n6)
(plus1 n6 n7)
(plus1 n7 n8)
(plus1 n8 n9)
(plus1 n9 n10)
```

```
(= (total-cost) 0)
(= (cost-of n1) 1)
(= (cost-of n2) 2)
(= (cost-of n3) 3)
(= (cost-of n4) 4)
(= (cost-of n5) 5)
(= (cost-of n6) 6)
(= (cost-of n7) 7)
(= (cost-of n8) 8)
(= (cost-of n9) 9)
(= (cost-of n10) 10)
```

```
(= (distance R1 R1) 0)
(= (distance R1 R2) 3)
(= (distance R1 R3) 2)
(= (distance R1 R4) 4)
(= (distance R1 R5) 5)
(= (distance R2 R1) 3)
(= (distance R2 R2) 0)
(= (distance R2 R3) 1)
(= (distance R2 R4) 3)
(= (distance R2 R5) 4)
(= (distance R3 R1) 2)
(= (distance R3 R2) 1)
(= (distance R3 R3) 0)
(= (distance R3 R4) 2)
(= (distance R3 R5) 3)
(= (distance R4 R1) 4)
(= (distance R4 R2) 3)
(= (distance R4 R3) 2)
(= (distance R4 R4) 0)
(= (distance R4 R5) 1)
(= (distance R5 R1) 5)
(= (distance R5 R2) 4)
(= (distance R5 R3) 3)
(= (distance R5 R4) 1)
(= (distance R5 R5) 0)
```

Problem instance 1: Results

There are three rooms and one robot in the environment. The robot already has all the keys and all the cleaning tools in the initial state. The robot needs to return to the room where it started (R1) after cleaning all the rooms.

Blind heuristic plan:

```
(move cleaningbot r1 r3)
(clean-mop cleaningbot r3 n9 mop)
(move cleaningbot r3 r2)
(sweep cleaningbot r2 n3 sweeper)
(clean-mop cleaningbot r2 n3 mop)
(move cleaningbot r2 r1)
(sweep cleaningbot r1 n6 sweeper)
; cost = 27 (general cost)
```

Expanded/Generated nodes: 48/126

Peak memory usage: 9856 KB

Execution time: 0.09s of which 0.001056s of search time

Problem instance 1: Results

There are three rooms and one robot in the environment. The robot already has all the keys and all the cleaning tools in the initial state. The robot needs to return to the room where it started (R1) after cleaning all the rooms.

h^{max} heuristic plan:

```
(sweep cleaningbot r1 n6 sweeper)
(move cleaningbot r1 r2)
(sweep cleaningbot r2 n3 sweeper)
(clean-mop cleaningbot r2 n3 mop)
(move cleaningbot r2 r3)
(clean-mop cleaningbot r3 n9 mop)
(move cleaningbot r3 r1)
; cost = 27 (general cost)
```

Expanded/Generated nodes: 40/109

Peak memory usage: 9856 KB

Execution time: 0.07s of which 0.000441s of search time

Problem instance 1: Results

There are three rooms and one robot in the environment. The robot already has all the keys and all the cleaning tools in the initial state. The robot needs to return to the room where it started (R1) after cleaning all the rooms.

h^{FF} heuristic plan:

```
(sweep cleaningbot r1 n6 sweeper)
(move cleaningbot r1 r3)
(clean-mop cleaningbot r3 n9 mop)
(move cleaningbot r3 r2)
(sweep cleaningbot r2 n3 sweeper)
(clean-mop cleaningbot r2 n3 mop)
(move cleaningbot r2 r1)
; cost = 27 (general cost)
```

Expanded/Generated nodes: 10/24

Peak memory usage: 9856 KB

Execution time: 0.07s of which 0.000264s of search time

Problem instances: 2

There are three rooms and one robot in the environment. The robot needs to look for the keys and the cleaning tools. The robot needs to return to the room where it started (R1) after cleaning all the rooms.

Objects:

```
(:objects
  R1 R2 R3 - room
  CleaningBot - robot
  n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 - quantity
  Key1 Key2 - key
  Mop - mop
  Sweeper - sweeper
)
```

Initial state:

```
(:init
  (at-robot CleaningBot R1)
  (dirty R1)
  (dirty R2)
  (dirty R3)
  (to-be-cleaned R1 n6)
  (to-be-cleaned R2 n3)
  (to-be-cleaned R3 n9)
  (locked R2)
  (locked R3)
  (key-opens Key2 R2)
  (key-opens Key1 R3)
  (at-key Key1 R1)
  (at-key Key2 R3)
  (at-mop Mop R2)
  (at-sweeper Sweeper R1)
  (has-dust R1)
  (has-dust R2)
  (has-stains R2)
  (has-stains R3)
)
```

Problem instance 2: Results

Blind heuristic plan:

```
(pick-up-key cleaningbot r1 key1)
(unlock cleaningbot r3 key1)
(pick-up-sweeper cleaningbot r1 sweeper)
(move cleaningbot r1 r3)
(pick-up-key cleaningbot r3 key2)
(unlock cleaningbot r2 key2)
(move cleaningbot r3 r2)
(pick-up-mop cleaningbot r2 mop)
(sweep cleaningbot r2 n3 sweeper)
(clean-mop cleaningbot r2 n3 mop)
(move cleaningbot r2 r3)
(clean-mop cleaningbot r3 n9 mop)
(move cleaningbot r3 r1)
(sweep cleaningbot r1 n6 sweeper)
; cost = 33 (general cost)
```

Expanded/Generated nodes: 92/231

Peak memory usage: 9856 KB

Execution time: 0.07s of which 0.000511s of search time

Problem instance 2: Results

h^{max} heuristic plan:

```
(pick-up-key cleaningbot r1 key1)
(unlock cleaningbot r3 key1)
(pick-up-sweeper cleaningbot r1 sweeper)
(sweep cleaningbot r1 n6 sweeper)
(move cleaningbot r1 r3)
(pick-up-key cleaningbot r3 key2)
(unlock cleaningbot r2 key2)
(move cleaningbot r3 r2)
(pick-up-mop cleaningbot r2 mop)
(sweep cleaningbot r2 n3 sweeper)
(clean-mop cleaningbot r2 n3 mop)
(move cleaningbot r2 r3)
(clean-mop cleaningbot r3 n9 mop)
(move cleaningbot r3 r1)
; cost = 33 (general cost)
```

Expanded/Generated nodes: 84/214

Peak memory usage: 9856 KB

Execution time: 0.08s of which 0.000651s of search time

Problem instance 2: Results

h^{FF} heuristic plan:

```
(pick-up-key cleaningbot r1 key1)
(unlock cleaningbot r3 key1)
(pick-up-sweeper cleaningbot r1 sweeper)
(sweep cleaningbot r1 n6 sweeper)
(move cleaningbot r1 r3)
(pick-up-key cleaningbot r3 key2)
(unlock cleaningbot r2 key2)
(move cleaningbot r3 r2)
(sweep cleaningbot r2 n3 sweeper)
(pick-up-mop cleaningbot r2 mop)
(clean-mop cleaningbot r2 n3 mop)
(move cleaningbot r2 r3)
(clean-mop cleaningbot r3 n9 mop)
(move cleaningbot r3 r1)
; cost = 33 (general cost)
```

Expanded/Generated nodes: 26/52

Peak memory usage: 9856 KB

Execution time: 0.08s of which 0.000511s of search time

Problem instances: 3

There are three rooms and two robots in the environment. The robots need to look for the keys and the cleaning tools. The robots need to go to a specific room (R2) after cleaning all the rooms.

Objects:

```
(:objects
  R1 R2 R3 - room
  CleaningBot1 CleaningBot2 - robot
  n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 - quantity
  Key1 Key2 - key
  Mop - mop
  Sweeper - sweeper
)
```

Initial state:

```
(:init
  (at-robot CleaningBot1 R1)
  (at-robot CleaningBot2 R3)
  (dirty R1)
  (dirty R2)
  (dirty R3)
  (to-be-cleaned R1 n6)
  (to-be-cleaned R2 n3)
  (to-be-cleaned R3 n9)
  (locked R2)
  (locked R3)
  (at-key Key1 R2)
  (at-key Key2 R3)
  (key-opens Key2 R2)
  (key-opens Key1 R3)
  (at-mop Mop R3)
  (at-sweeper Sweeper R1)
  (has-dust R1)
  (has-dust R2)
  (has-dust R3)
  (has-stains R1)
  (has-stains R2)
  (has-stains R3)
)
```

Problem instance 3: Results

Blind heuristic plan:

```
(pick-up-key cleaningbot2 r3 key2)
(unlock cleaningbot2 r2 key2)
(move cleaningbot2 r3 r2)
(pick-up-key cleaningbot2 r2 key1)
(unlock cleaningbot2 r3 key1)
(pick-up-sweeper cleaningbot1 r1 sweeper)
(move cleaningbot1 r1 r3)
(pick-up-mop cleaningbot1 r3 mop)
(sweep cleaningbot1 r3 n9 sweeper)
(clean-mop cleaningbot1 r3 n9 mop)
(move cleaningbot1 r3 r1)
(sweep cleaningbot1 r1 n6 sweeper)
(clean-mop cleaningbot1 r1 n6 mop)
(move cleaningbot1 r1 r2)
(sweep cleaningbot1 r2 n3 sweeper)
(clean-mop cleaningbot1 r2 n3 mop)
; cost = 50 (general cost)
```

Expanded/Generated nodes: 6097/29557

Peak memory usage: 10124 KB

Execution time: 0.08s of which 0.009916s of search time

Problem instance 3: Results

h^{max} heuristic plan:

```
(pick-up-mop cleaningbot2 r3 mop)
(pick-up-key cleaningbot2 r3 key2)
(unlock cleaningbot2 r2 key2)
(clean-mop cleaningbot2 r3 n9 mop)
(move cleaningbot2 r3 r1)
(clean-mop cleaningbot2 r1 n6 mop)
(move cleaningbot2 r1 r2)
(pick-up-key cleaningbot2 r2 key1)
(unlock cleaningbot2 r3 key1)
(pick-up-sweeper cleaningbot1 r1 sweeper)
(sweep cleaningbot1 r1 n6 sweeper)
(move cleaningbot1 r1 r3)
(clean-mop cleaningbot2 r2 n3 mop)
(sweep cleaningbot1 r3 n9 sweeper)
(move cleaningbot1 r3 r2)
(sweep cleaningbot1 r2 n3 sweeper)
; cost = 50 (general cost)
```

Expanded/Generated nodes: 4954/24472

Peak memory usage: 10140 KB

Execution time: 0.09s of which 0.013140s of search time

Problem instance 3: Results

h^{FF} heuristic plan:

```
(pick-up-key cleaningbot2 r3 key2)
(unlock cleaningbot2 r2 key2)
(move cleaningbot2 r3 r2)
(pick-up-key cleaningbot2 r2 key1)
(unlock cleaningbot2 r3 key1)
(pick-up-sweeper cleaningbot1 r1 sweeper)
(move cleaningbot1 r1 r3)
(sweep cleaningbot1 r3 n9 sweeper)
(pick-up-mop cleaningbot1 r3 mop)
(clean-mop cleaningbot1 r3 n9 mop)
(move cleaningbot1 r3 r1)
(sweep cleaningbot1 r1 n6 sweeper)
(clean-mop cleaningbot1 r1 n6 mop)
(move cleaningbot1 r1 r2)
(sweep cleaningbot1 r2 n3 sweeper)
(clean-mop cleaningbot1 r2 n3 mop)
; cost = 50 (general cost)
```

Expanded/Generated nodes: 103/331

Peak memory usage: 9856 KB

Execution time: 0.07s of which 0.001096s of search time

Problem instances: 4

There are five rooms and one robot in the environment. The robot needs to look for the keys and the cleaning tools. The robot needs to return to the room where it started (R1) after cleaning all the rooms.

Objects:

```
(:objects
  R1 R2 R3 R4 R5 - room
  CleaningBot - robot
  n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 - quantity
  Key1 Key2 Key3 - key
  Mop - mop
  Sweeper - sweeper
)
```

Initial state:

```
(:init
  (at-robot CleaningBot R1)
  (dirty R1)
  (dirty R2)
  (dirty R3)
  (dirty R4)
  (dirty R5)
  (to-be-cleaned R1 n6)
  (to-be-cleaned R2 n3)
  (to-be-cleaned R3 n9)
  (to-be-cleaned R4 n5)
  (to-be-cleaned R5 n10)
  (locked R2)
  (locked R3)
  (locked R4)
  (at-key Key1 R1)
  (at-key Key2 R3)
  (at-key Key3 R5)
  (key-opens Key2 R2)
  (key-opens Key1 R3)
  (key-opens Key3 R4)
  (at-mop Mop R2)
  (at-sweeper Sweeper R1)
  (has-dust R1)
  (has-dust R2)
  (has-dust R4)
  (has-stains R2)
  (has-stains R3)
  (has-stains R5)
)
```

Problem instance 4: Results

Blind heuristic plan:

```
(pick-up-key cleaningbot r1 key1)
(unlock cleaningbot r3 key1)
(pick-up-sweeper cleaningbot r1 sweeper)
(move cleaningbot r1 r3)
(pick-up-key cleaningbot r3 key2)
(unlock cleaningbot r2 key2)
(move cleaningbot r3 r2)
(pick-up-mop cleaningbot r2 mop)
(sweep cleaningbot r2 n3 sweeper)
(clean-mop cleaningbot r2 n3 mop)
(move cleaningbot r2 r3)
(clean-mop cleaningbot r3 n9 mop)
(move cleaningbot r3 r5)
(pick-up-key cleaningbot r5 key3)
(unlock cleaningbot r4 key3)
(clean-mop cleaningbot r5 n10 mop)
(move cleaningbot r5 r4)
(sweep cleaningbot r4 n5 sweeper)
(move cleaningbot r4 r1)
(sweep cleaningbot r1 n6 sweeper)
; cost = 56 (general cost)
```

Expanded/Generated nodes: 894/3812

Peak memory usage: 9856 KB

Execution time: 0.08s of which 0.001847s of search time

Problem instance 4: Results

h^{max} heuristic plan:

```
(pick-up-key cleaningbot r1 key1)
(unlock cleaningbot r3 key1)
(pick-up-sweeper cleaningbot r1 sweeper)
(sweep cleaningbot r1 n6 sweeper)
(move cleaningbot r1 r3)
(pick-up-key cleaningbot r3 key2)
(unlock cleaningbot r2 key2)
(move cleaningbot r3 r2)
(pick-up-mop cleaningbot r2 mop)
(sweep cleaningbot r2 n3 sweeper)
(clean-mop cleaningbot r2 n3 mop)
(move cleaningbot r2 r5)
(pick-up-key cleaningbot r5 key3)
(unlock cleaningbot r4 key3)
(clean-mop cleaningbot r5 n10 mop)
(move cleaningbot r5 r4)
(sweep cleaningbot r4 n5 sweeper)
(move cleaningbot r4 r3)
(clean-mop cleaningbot r3 n9 mop)
(move cleaningbot r3 r1)
; cost = 56 (general cost)
```

Expanded/Generated nodes: 851/3643

Peak memory usage: 9856 KB

Execution time: 0.06s of which 0.003793s of search time

Problem instance 4: Results

h^{FF} heuristic plan:

```
(pick-up-key cleaningbot r1 key1)
(unlock cleaningbot r3 key1)
(pick-up-sweeper cleaningbot r1 sweeper)
(sweep cleaningbot r1 n6 sweeper)
(move cleaningbot r1 r3)
(pick-up-key cleaningbot r3 key2)
(unlock cleaningbot r2 key2)
(move cleaningbot r3 r2)
(sweep cleaningbot r2 n3 sweeper)
(pick-up-mop cleaningbot r2 mop)
(clean-mop cleaningbot r2 n3 mop)
(move cleaningbot r2 r3)
(clean-mop cleaningbot r3 n9 mop)
(move cleaningbot r3 r5)
(clean-mop cleaningbot r5 n10 mop)
(pick-up-key cleaningbot r5 key3)
(unlock cleaningbot r4 key3)
(move cleaningbot r5 r4)
(sweep cleaningbot r4 n5 sweeper)
(move cleaningbot r4 r1)
; cost = 56 (general cost)
```

Expanded/Generated nodes: 37/125

Peak memory usage: 9856 KB

Execution time: 0.07s of which 0.000665s of search time

Problem instances: 5

There are five rooms and two robots in the environment. The robots need to look for the keys and the cleaning tools. The robots need to go to a specific room after cleaning all the rooms.

Objects:

```
(:objects
  R1 R2 R3 R4 R5 - room
  CleaningBot1 CleaningBot2 - robot
  n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 - quantity
  Key1 Key2 Key3 - key
  Mop - mop
  Sweeper - sweeper
)
```

Initial state:

```
(:init
  (at-robot CleaningBot1 R5)
  (at-robot CleaningBot2 R3)
  (dirty R1)
  (dirty R2)
  (dirty R3)
  (dirty R4)
  (dirty R5)
  (to-be-cleaned R1 n6)
  (to-be-cleaned R2 n3)
  (to-be-cleaned R3 n9)
  (to-be-cleaned R4 n5)
  (to-be-cleaned R5 n10)
  (locked R2)
  (locked R3)
  (locked R5)
  (at-key Key1 R2)
  (at-key Key2 R3)
  (at-key Key3 R1)
  (key-opens Key2 R2)
  (key-opens Key1 R3)
  (key-opens Key3 R5)
  (at-mop Mop R1)
  (at-sweeper Sweeper R5)
  (has-dust R1)
  (has-dust R2)
  (has-dust R4)
  (has-stains R2)
  (has-stains R3)
  (has-stains R5)
)
```

Problem instance 5: Results

Blind heuristic plan:

```
(pick-up-key cleaningbot2 r3 key2)
(unlock cleaningbot2 r2 key2)
(pick-up-sweeper cleaningbot1 r5 sweeper)
(move cleaningbot1 r5 r4)
(sweep cleaningbot1 r4 n5 sweeper)
(move cleaningbot1 r4 r1)
(pick-up-key cleaningbot1 r1 key3)
(unlock cleaningbot1 r5 key3)
(pick-up-mop cleaningbot1 r1 mop)
(move cleaningbot1 r1 r2)
(pick-up-key cleaningbot1 r2 key1)
(unlock cleaningbot1 r3 key1)
(sweep cleaningbot1 r2 n3 sweeper)
(clean-mop cleaningbot1 r2 n3 mop)
(move cleaningbot1 r2 r3)
(move cleaningbot2 r3 r1)
(clean-mop cleaningbot1 r3 n9 mop)
(move cleaningbot1 r3 r5)
(clean-mop cleaningbot1 r5 n10 mop)
(move cleaningbot1 r5 r1)
(sweep cleaningbot1 r1 n6 sweeper)
; cost = 63 (general cost)
```

Peak memory usage:12020KB

Expanded/Generated nodes: 56208/443027

Execution time: 0.14s of which 0.070903s of search time

Problem instance 5: Results

h^{max} heuristic plan:

```
(pick-up-key cleaningbot2 r3 key2)
(unlock cleaningbot2 r2 key2)
(pick-up-sweeper cleaningbot1 r5 sweeper)
(move cleaningbot1 r5 r1)
(pick-up-key cleaningbot1 r1 key3)
(unlock cleaningbot1 r5 key3)
(pick-up-mop cleaningbot1 r1 mop)
(sweep cleaningbot1 r1 n6 sweeper)
(move cleaningbot1 r1 r5)
(clean-mop cleaningbot1 r5 n10 mop)
(move cleaningbot1 r5 r4)
(sweep cleaningbot1 r4 n5 sweeper)
(move cleaningbot1 r4 r2)
(pick-up-key cleaningbot1 r2 key1)
(unlock cleaningbot1 r3 key1)
(sweep cleaningbot1 r2 n3 sweeper)
(clean-mop cleaningbot1 r2 n3 mop)
(move cleaningbot1 r2 r3)
(clean-mop cleaningbot1 r3 n9 mop)
(move cleaningbot1 r3 r1)
(move cleaningbot2 r3 r1)
; cost = 63 (general cost)
```

Peak memory usage:11868KB

Expanded/Generated nodes: 48338/383511

Execution time: 0.19s of which 0.120517s of search time

Problem instance 5: Results

h^{FF} heuristic plan:

```
(pick-up-sweeper cleaningbot1 r5 sweeper)
(pick-up-key cleaningbot2 r3 key2)
(unlock cleaningbot2 r2 key2)
(move cleaningbot1 r5 r4)
(sweep cleaningbot1 r4 n5 sweeper)
(move cleaningbot1 r4 r1)
(sweep cleaningbot1 r1 n6 sweeper)
(pick-up-key cleaningbot1 r1 key3)
(unlock cleaningbot1 r5 key3)
(move cleaningbot2 r3 r1)
(pick-up-mop cleaningbot1 r1 mop)
(move cleaningbot1 r1 r2)
(sweep cleaningbot1 r2 n3 sweeper)
(clean-mop cleaningbot1 r2 n3 mop)
(pick-up-key cleaningbot1 r2 key1)
(unlock cleaningbot1 r3 key1)
(move cleaningbot1 r2 r3)
(clean-mop cleaningbot1 r3 n9 mop)
(move cleaningbot1 r3 r5)
(clean-mop cleaningbot1 r5 n10 mop)
(move cleaningbot1 r5 r1)
; cost = 63 (general cost)
```

Peak memory usage: 10124KB

Expanded/Generated nodes: 785/5764

Execution time: 0.07s of which 0.008145s of search time

Thank you for your attention!

SAPIENZA
UNIVERSITÀ DI ROMA