TDDC17
Artificial Intelligence

Lab 4

Cristopher Dahlström crida498@student.liu.se
Jonathan Lundholm jonlu159@student.liu.se

Task 1

The chosen domain is: Shakey's world.

The following assumptions for the world have been made:

- There is always one, and only one, light switch in each room
- Shakey kan push a box through a door without entering the room himself
- There is only one Shakey

Objects and predicates

Objects and predicates		Comment
ITEM	?item	Small item that can be picked up
GRIP	?grip	Griphook used to pickup items with
ROOM	?room	Location
DOOR	?room1 ?room2	Connects two locations
WIDE-DOOR	?room1 ?room2	Connects two locations; can push boxes through it
BOX	?box	Pushable object, required for turning lights on
in-room	?room	Defines which room Shakey is in
box-location	?room ?box	Defines the locations of a specific box
is-lit	?room	Defines whether a room is lit
gripped-item	?item ?grip	Defines whether an item is gripped
item-location	?item ?room	Defines the location of an item
used	?grip	Defines whether a gripper is used

Operators

Action	Properties	Comment
move	?from ?to	Moves Shakey from one room to another
move_box	?box ?from ?to	Moves a specific box from one room to another
turn_light_on	?room ?box	Turns the lights on in a room if there is a box to stand on
turn_light_off	?room ?box	Turns the lights off in a room if there is a box to stand on
grip	?room ?item ?grip	Grips an item with a gripper
drop	?room ?item ?grip	Drops an item with a gripper

All operators except for switching the light on and off are self-explanatory. The prerequisite for being able to turn lights on or off is that Shakey stands on a box, but since switching the lights on or off always

includes standing on a box, this functionality have been merged into the turn_light_on and turn_light_off actions.

Task 2

The methodology for testing the performance is the following.

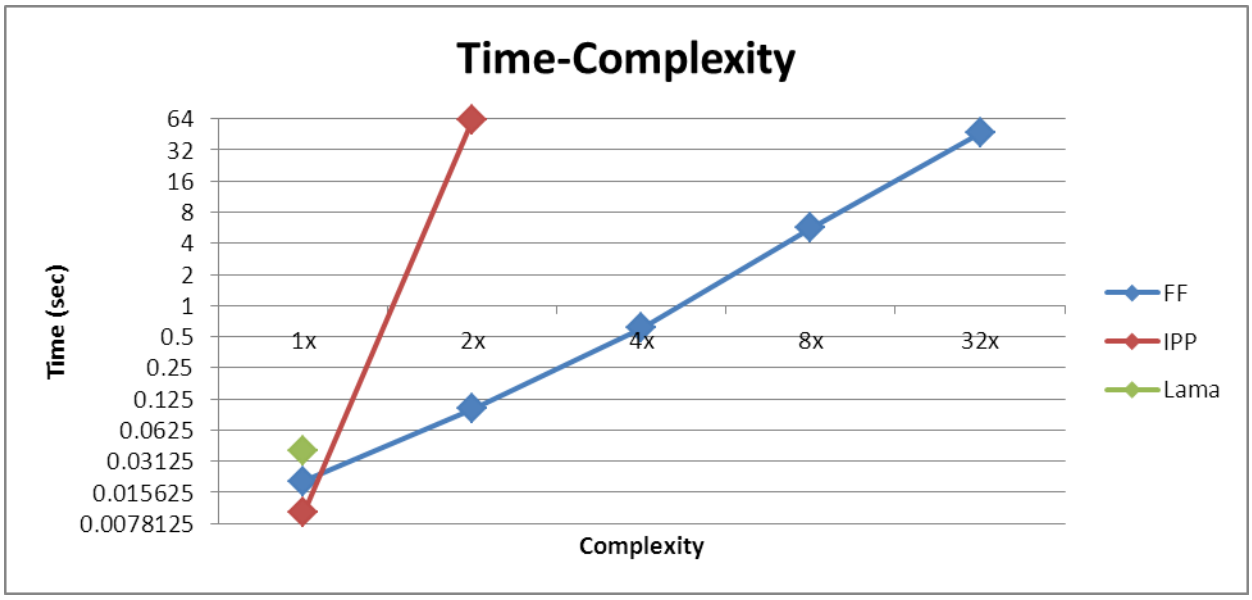
The initial test was done with a fairly small domain and set of goals that checks whether all different actions Shakey can perform are carried out correctly; this domain is then copied and connected.

For the second test another identical domain was added, with the same properties, connected to the already existing domains with a door; these two domains combine into a bigger domain with higher complexity. This procedure is repeated, thus forming our test-suite.

Solver	Domain complexity	Steps	Time (sec)
FF	1x	14	0.00
FF	2x	32	0.02
FF	4x	50	0.10
FF	8x	69	0.61
FF	16x	143	5.63
FF	32x	291	46.47
FF	64x	Out of memory	
IPP	1x	11	0.01
IPP	2x	25	62.52
IPP	4x	Aborted due being +10 min	
Lama	1x	16	0.04
Lama	2x	Fails to run	

One possible reason for how well the planners manage to handle the scaled up problems is how locally they work. Due to the structure, the added complexity still only adds new areas with similar problems, ergo, an ideal solver for this problem would solve the first part locally, move on to the next and solve that locally, and so on. Any solver that attempts to find an “ultimate solution” or a solution that ranges over all different parts of the area is bound to run into problems fast.

If plotting the table above, using a logarithmic scale (base 2), one sees that the time increase appears to be fairly linear; this is by no means surprising due to the reasoning about complexity above. Lama does not appear in the chart due to being unable to solve the problem at any other complexity than x1.



The following graph shows how the complexity relates to the number of steps required to solve the problem. If extrapolating straight lines from the FF and the IPP we notice that IPP seems to have slightly better solutions; this connects to the reasoning about the increased complexity as well; the planner solves the problem in a more optimal manner, at the cost of operating time, but due to this the complexity increases at a factor of 2, while the complexity of FF increases with a small factor, probably due to relying, to a greater degree, on heuristics and non-optimal solutions.