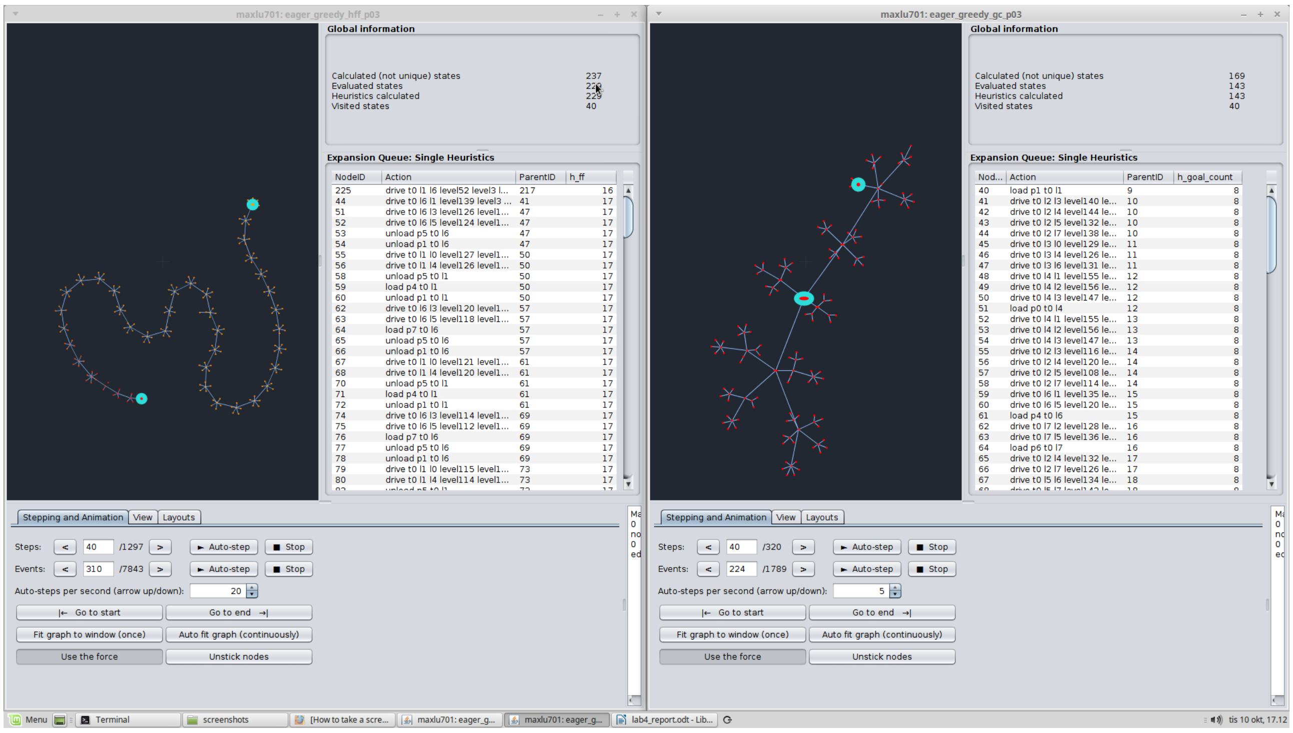
TDDC17 Lab 3

Max Lund, maxlu701
Joakim Strandberg, joast388

1. Comparing heuristics, problem 03

We can see in the initial steps of the two planners that they begin by expanding nodes with different actions right after the initial node is expanded. The exact difference between these actions are different to interpret, but stepping through the first 40 nodes in both planners, we can see that the FF-heuristic produces mostly drives with some loads and unloads, while the goal-count heuristic produces mostly drives with a few loads and no unloads.

Looking at the first 40 steps of the two planners, we can see that the goal-count heuristic produces what looks like a BFS algorithm, while the FF-heuristic performs a search through the nodes which have the lowest cost (i.e. a greedy search). All nodes in the goal count heuristic has the same heuristic value and therfore it behaves as a BFS-search. The FF-heuristic has different cost for child-nodes, so it ”traverses” (expands nodes) through a single path of the children that has the lowest cost.



Global information

Calculated (not unique) states	237
Evaluated states	229
Heuristics calculated	229
Visited states	40

Expansion Queue: Single Heuristics

NodeID	Action	ParentID	h_ff
225	drive t0 l1 l6 level52 level3 l...	217	16
44	drive t0 l6 l1 level139 level3 ...	41	17
51	drive t0 l6 l3 level126 level1...	47	17
52	drive t0 l6 l5 level124 level1...	47	17
53	unload p5 t0 l6	47	17
54	unload p1 t0 l6	47	17
55	drive t0 l1 l0 level127 level1...	50	17
56	drive t0 l1 l4 level126 level1...	50	17
58	unload p5 t0 l1	50	17
59	load p4 t0 l1	50	17
60	unload p1 t0 l1	50	17
62	drive t0 l6 l3 level120 level1...	57	17
63	drive t0 l6 l5 level118 level1...	57	17
64	load p7 t0 l6	57	17
65	unload p5 t0 l6	57	17
66	unload p1 t0 l6	57	17
67	drive t0 l1 l0 level121 level1...	61	17
68	drive t0 l1 l4 level120 level1...	61	17
70	unload p5 t0 l1	61	17
71	load p4 t0 l1	61	17
72	unload p1 t0 l1	61	17
74	drive t0 l6 l3 level114 level1...	69	17
75	drive t0 l6 l5 level112 level1...	69	17
76	load p7 t0 l6	69	17
77	unload p5 t0 l6	69	17
78	unload p1 t0 l6	69	17
79	drive t0 l1 l0 level115 level1...	73	17
80	drive t0 l1 l4 level114 level1...	73	17

Steps: 40 /1297
Events: 310 /7843
Auto-steps per second (arrow up/down): 20

Global information

Calculated (not unique) states	169
Evaluated states	143
Heuristics calculated	143
Visited states	40

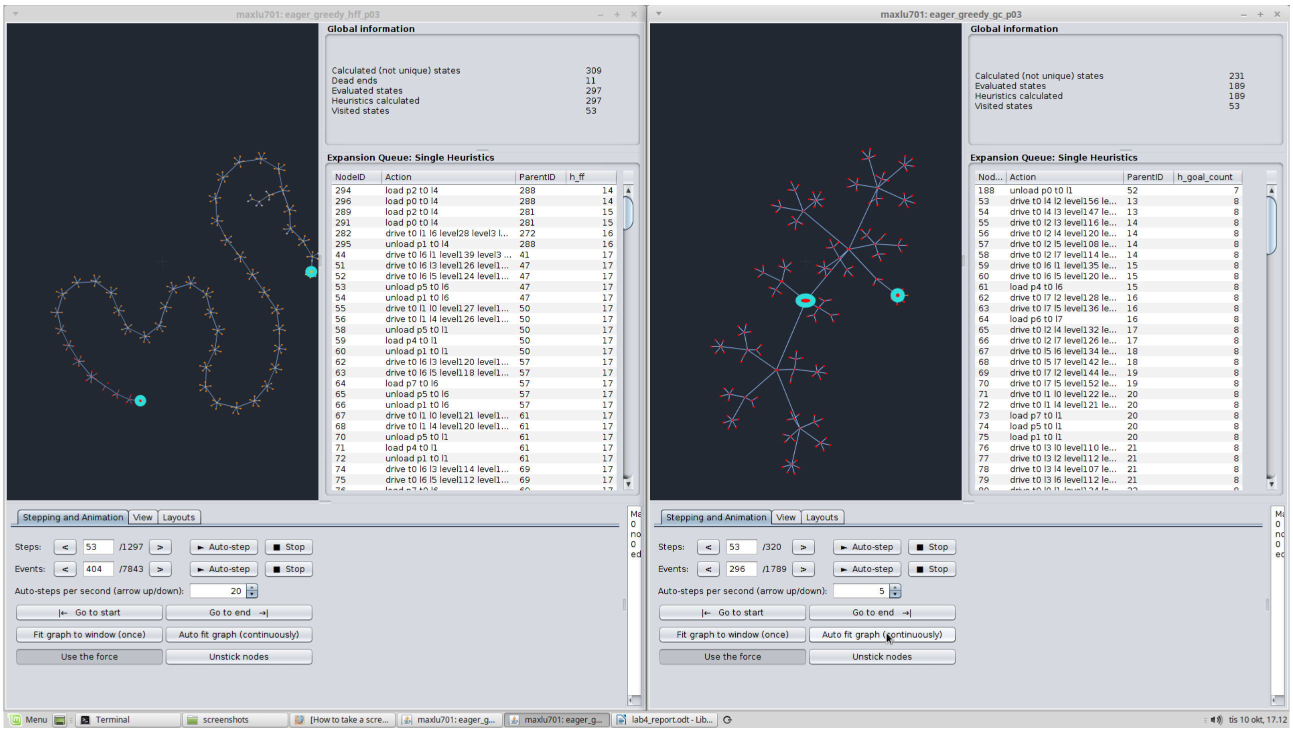
Expansion Queue: Single Heuristics

Nod...	Action	ParentID	h_goal_count
40	load p1 t0 l1	9	8
41	drive t0 l2 l3 level140 le...	10	8
42	drive t0 l2 l4 level144 le...	10	8
43	drive t0 l2 l5 level132 le...	10	8
44	drive t0 l2 l7 level138 le...	10	8
45	drive t0 l3 l0 level129 le...	11	8
46	drive t0 l3 l4 level126 le...	11	8
47	drive t0 l3 l6 level131 le...	11	8
48	drive t0 l4 l1 level155 le...	12	8
49	drive t0 l4 l2 level156 le...	12	8
50	drive t0 l4 l3 level147 le...	12	8
51	load p0 t0 l4	12	8
52	drive t0 l4 l1 level155 le...	13	8
53	drive t0 l4 l2 level156 le...	13	8
54	drive t0 l4 l3 level147 le...	13	8
55	drive t0 l2 l3 level116 le...	14	8
56	drive t0 l2 l4 level120 le...	14	8
57	drive t0 l2 l5 level108 le...	14	8
58	drive t0 l2 l7 level114 le...	14	8
59	drive t0 l6 l1 level135 le...	15	8
60	drive t0 l6 l5 level120 le...	15	8
61	load p4 t0 l6	15	8
62	drive t0 l7 l2 level128 le...	16	8
63	drive t0 l7 l5 level136 le...	16	8
64	load p6 t0 l7	16	8
65	drive t0 l2 l4 level132 le...	17	8
66	drive t0 l2 l7 level126 le...	17	8
67	drive t0 l5 l6 level134 le...	18	8

Steps: 40 /320
Events: 224 /1789
Auto-steps per second (arrow up/down): 5

Planners at 40 steps.

Looking at the intial expansions of the two planners, we can see that the one using the goal-count heuristic, the cost of expanding different nodes doesn't change intially (it stays at 8). The planner using the greedy search heuristic has different cost for the nodes, and expands the ones with lowest cost. From seeing this, we reason that there might be a lot of randomness while using the goal-count heuristic, since it can take many expansions before there is any difference in cost between nodes, i.e. when a node that reduces the goal-count appears.


maxlu701: eager_greedy_hff_p03
Stepping and Animation
View
Layouts
Auto-step
Stop
Go to start
Go to end
Fit graph to window (once)
Auto fit graph (continuously)
Use the force
Unstick nodes
maxlu701: eager_greedy_gc_p03

maxlu701: eager_greedy_hff_p03

Global information

```
Calculated (not unique) states          309
Dead ends                               11
Evaluated states                        297
Heuristics calculated                   297
Visited states                          53
```

Expansion Queue: Single Heuristics

NodeID	Action	ParentID	h_ff
294	load p2 t0 l4	288	14
296	load p0 t0 l4	288	14
289	load p2 t0 l4	281	15
291	load p0 t0 l4	281	15
282	drive t0 l1 l6 level28 level3 l...	272	16
295	unload p1 t0 l4	288	16
44	drive t0 l6 l1 level139 level3 ...	41	17
51	drive t0 l6 l3 level126 level1...	47	17
52	drive t0 l6 l5 level124 level1...	47	17
53	unload p5 t0 l6	47	17
54	unload p1 t0 l6	47	17
55	drive t0 l1 l0 level127 level1...	50	17
56	drive t0 l1 l4 level126 level1...	50	17
58	unload p5 t0 l1	50	17
59	load p4 t0 l1	50	17
60	unload p1 t0 l1	50	17
62	drive t0 l6 l3 level120 level1...	57	17
63	drive t0 l6 l5 level118 level1...	57	17
64	load p7 t0 l6	57	17
65	unload p5 t0 l6	57	17
66	unload p1 t0 l6	57	17
67	drive t0 l1 l0 level121 level1...	61	17
68	drive t0 l1 l4 level120 level1...	61	17
70	unload p5 t0 l1	61	17
71	load p4 t0 l1	61	17
72	unload p1 t0 l1	61	17
74	drive t0 l6 l3 level114 level1...	69	17
75	drive t0 l6 l5 level112 level1...	69	17

Steps: 53 /1297
Events: 404 /7843
Auto-steps per second (arrow up/down): 20

maxlu701: eager_greedy_gc_p03

Global information

```
Calculated (not unique) states          231
Evaluated states                        189
Heuristics calculated                   189
Visited states                          53
```

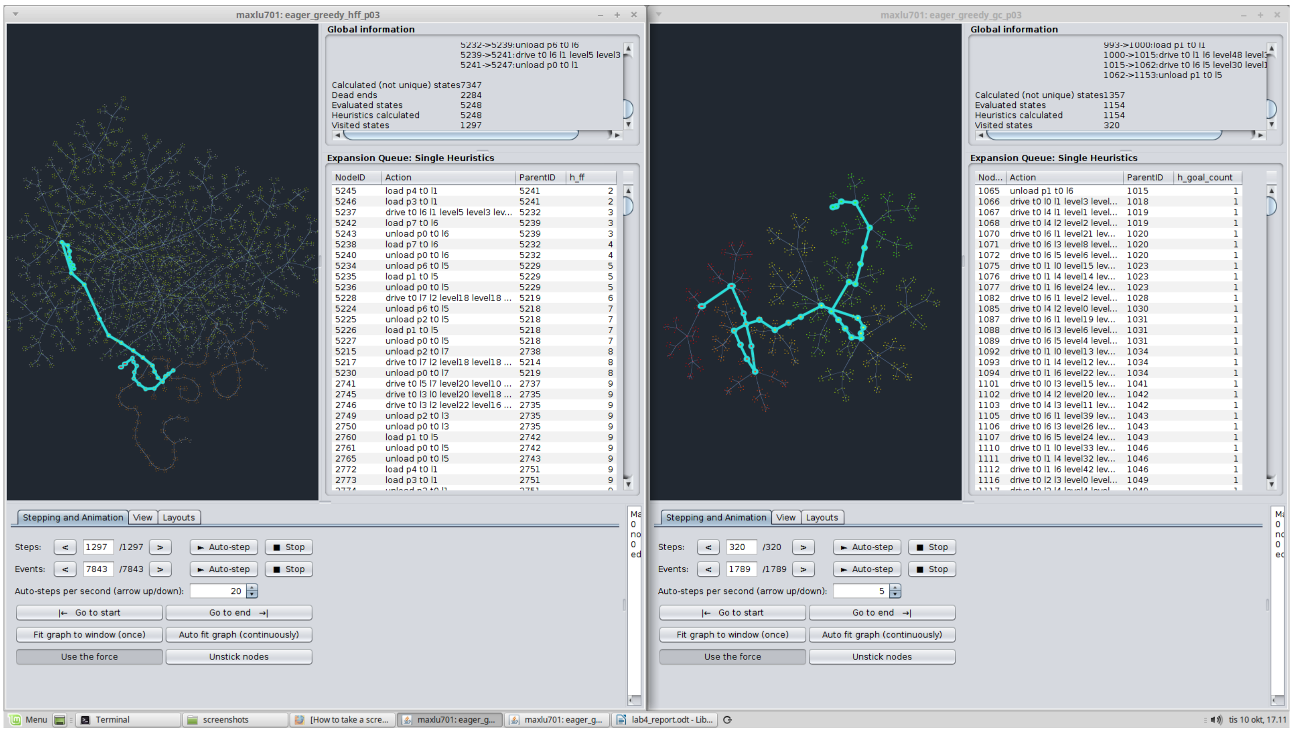
Expansion Queue: Single Heuristics

Nod...	Action	ParentID	h_goal_count
188	unload p0 t0 l1	52	7
53	drive t0 l4 l2 level156 le...	13	8
54	drive t0 l4 l3 level147 le...	13	8
55	drive t0 l2 l3 level116 le...	14	8
56	drive t0 l2 l4 level120 le...	14	8
57	drive t0 l2 l5 level108 le...	14	8
58	drive t0 l2 l7 level114 le...	14	8
59	drive t0 l6 l1 level135 le...	15	8
60	drive t0 l6 l5 level120 le...	15	8
61	load p4 t0 l6	15	8
62	drive t0 l7 l2 level128 le...	16	8
63	drive t0 l7 l5 level136 le...	16	8
64	load p6 t0 l7	16	8
65	drive t0 l2 l4 level132 le...	17	8
66	drive t0 l2 l7 level126 le...	17	8
67	drive t0 l5 l6 level134 le...	18	8
68	drive t0 l5 l7 level142 le...	18	8
69	drive t0 l7 l2 level144 le...	19	8
70	drive t0 l7 l5 level152 le...	19	8
71	drive t0 l1 l0 level122 le...	20	8
72	drive t0 l1 l4 level121 le...	20	8
73	load p7 t0 l1	20	8
74	load p5 t0 l1	20	8
75	load p1 t0 l1	20	8
76	drive t0 l3 l0 level110 le...	21	8
77	drive t0 l3 l2 level112 le...	21	8
78	drive t0 l3 l4 level107 le...	21	8
79	drive t0 l3 l6 level112 le...	21	8

Steps: 53 /320
Events: 296 /1789
Auto-steps per second (arrow up/down): 5

The first goal-count lower than 8 appears after 53 steps.

In the end, we can see that intial ”snake” generated by the FF-heuristic search was not used in the final path to the goal. Furthermore, this heuristic obviously used a lot more nodes to find a goal-path. The reason for this is that, just as any greedy-search algorithm, it can favor a path which might look promising but will not bring you closer to your actual goal, but rather into a ”dead-end”. While the goal-count heuristic also uses greedy search, it only goes down a path that is actually guaranteed to accomplish a goal, not just a path which might look promising. Perhaps it is possible to generate nodes which gets the goal-count down to 1, but still ends up being a dead-end, but from these examples, it seems that using goal-count as your heuristic in a greedy-search has a higher probability of actually getting you closer to a goal-path.


maxlu701: eager_greedy_hff_p03
Stepping and Animation
View
Layouts
Auto-step
Stop
Go to start
Go to end
Fit graph to window (once)
Auto fit graph (continuously)
Use the force
Unstick nodes
maxlu701: eager_greedy_gc_p03

maxlu701: eager_greedy_hff_p03

Global information

```
                                        5232->5239:unload p6 t0 l6
                                        5239->5241:drive t0 l6 l1 level5 level3
                                        5241->5247:unload p0 t0 l1

Calculated (not unique) states7347
Dead ends                     2284
Evaluated states              5248
Heuristics calculated         5248
Visited states                1297
```

Expansion Queue: Single Heuristics

NodeID	Action	ParentID	h_ff
5245	load p4 t0 l1	5241	2
5246	load p3 t0 l1	5241	2
5237	drive t0 l6 l1 level5 level3 lev...	5232	3
5242	load p7 t0 l6	5239	3
5243	unload p0 t0 l6	5239	3
5238	load p7 t0 l6	5232	4
5240	unload p0 t0 l6	5232	4
5234	unload p6 t0 l5	5229	5
5235	load p1 t0 l5	5229	5
5236	unload p0 t0 l5	5229	5
5228	drive t0 l7 l2 level18 level18 ...	5219	6
5224	unload p6 t0 l5	5218	7
5225	unload p2 t0 l5	5218	7
5226	load p1 t0 l5	5218	7
5227	unload p0 t0 l5	5218	7
5215	unload p2 t0 l7	2738	8
5217	drive t0 l7 l2 level18 level18 ...	5214	8
5230	unload p0 t0 l7	5219	8
2741	drive t0 l5 l7 level20 level10 ...	2737	9
2745	drive t0 l3 l0 level20 level18 ...	2735	9
2746	drive t0 l3 l2 level22 level16 ...	2735	9
2749	unload p2 t0 l3	2735	9
2750	unload p0 t0 l3	2735	9
2760	load p1 t0 l5	2742	9
2761	unload p0 t0 l5	2742	9
2765	unload p0 t0 l5	2743	9
2772	load p4 t0 l1	2751	9
2773	load p3 t0 l1	2751	9

Steps: 1297 /1297
Events: 7843 /7843
Auto-steps per second (arrow up/down): 20

maxlu701: eager_greedy_gc_p03

Global information

```
                                        993->1000:load p1 t0 l1
                                        1000->1015:drive t0 l1 l6 level48 level
                                        1015->1062:drive t0 l6 l5 level30 level
                                        1062->1153:unload p1 t0 l5

Calculated (not unique) states1357
Evaluated states              1154
Heuristics calculated         1154
Visited states                320
```

Expansion Queue: Single Heuristics

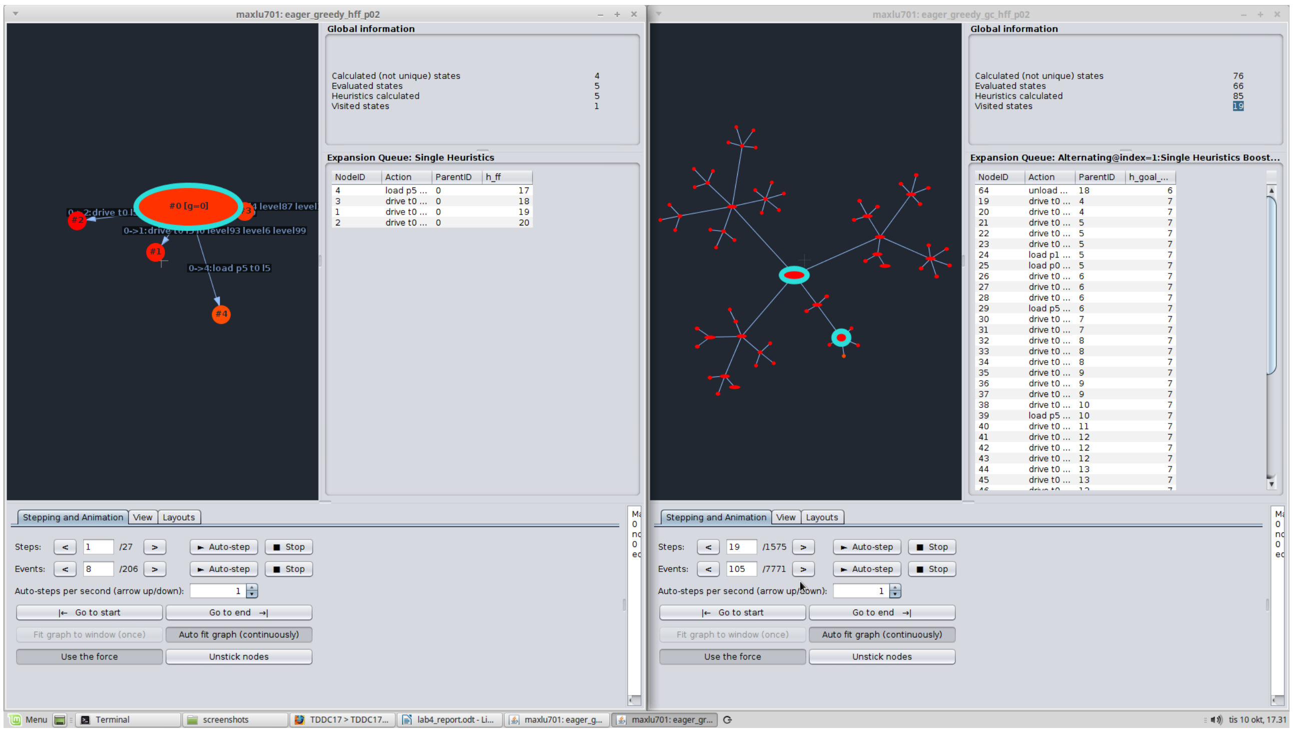
Nod...	Action	ParentID	h_goal_count
1065	unload p1 t0 l6	1015	1
1066	drive t0 l0 l1 level3 level...	1018	1
1067	drive t0 l4 l1 level1 level...	1019	1
1068	drive t0 l4 l2 level2 level...	1019	1
1070	drive t0 l6 l1 level21 lev...	1020	1
1071	drive t0 l6 l3 level8 level...	1020	1
1072	drive t0 l6 l5 level6 level...	1020	1
1075	drive t0 l1 l0 level15 lev...	1023	1
1076	drive t0 l1 l4 level14 lev...	1023	1
1077	drive t0 l1 l6 level24 lev...	1023	1
1082	drive t0 l6 l1 level2 level...	1028	1
1085	drive t0 l4 l2 level0 level...	1030	1
1087	drive t0 l6 l1 level19 lev...	1031	1
1088	drive t0 l6 l3 level6 level...	1031	1
1089	drive t0 l6 l5 level4 level...	1031	1
1092	drive t0 l1 l0 level13 lev...	1034	1
1093	drive t0 l1 l4 level12 lev...	1034	1
1094	drive t0 l1 l6 level22 lev...	1034	1
1101	drive t0 l0 l3 level15 lev...	1041	1
1102	drive t0 l4 l2 level20 lev...	1042	1
1103	drive t0 l4 l3 level11 lev...	1042	1
1105	drive t0 l6 l1 level39 lev...	1043	1
1106	drive t0 l6 l3 level26 lev...	1043	1
1107	drive t0 l6 l5 level24 lev...	1043	1
1110	drive t0 l1 l0 level33 lev...	1046	1
1111	drive t0 l1 l4 level32 lev...	1046	1
1112	drive t0 l1 l6 level42 lev...	1046	1
1116	drive t0 l2 l3 level0 level...	1049	1

Steps: 320 /320
Events: 1789 /1789
Auto-steps per second (arrow up/down): 5

Final paths to goal.

2. Comparing heuristics, problem 02

The FF-algorithm finds a node that reduces the heuristic value after the first expansion, time step1. The combined GC-FF-algorithm has to expand 19 nodes until the heuristic value is reduced from 7 to 6.


maxlu701: eager_greedy_hff_p02

Global information

```
Calculated (not unique) states     4
Evaluated states                   5
Heuristics calculated              5
Visited states                     1
```

Expansion Queue: Single Heuristics

NodeID	Action	ParentID	h_ff
4	load p5 ...	0	17
3	drive t0 ...	0	18
1	drive t0 ...	0	19
2	drive t0 ...	0	20

```
Steps:  1 /27
Events: 8 /206
Auto-steps per second (arrow up/down): 1
```

maxlu701: eager_greedy_gc_hff_p02

Global information

```
Calculated (not unique) states     76
Evaluated states                   66
Heuristics calculated              85
Visited states                     19
```

Expansion Queue: Alternating@index=1:Single Heuristics Boost...

NodeID	Action	ParentID	h_goal_...
64	unload ...	18	6
19	drive t0 ...	4	7
20	drive t0 ...	4	7
21	drive t0 ...	5	7
22	drive t0 ...	5	7
23	drive t0 ...	5	7
24	load p1 ...	5	7
25	load p0 ...	5	7
26	drive t0 ...	6	7
27	drive t0 ...	6	7
28	drive t0 ...	6	7
29	load p5 ...	6	7
30	drive t0 ...	7	7
31	drive t0 ...	7	7
32	drive t0 ...	8	7
33	drive t0 ...	8	7
34	drive t0 ...	8	7
35	drive t0 ...	9	7
36	drive t0 ...	9	7
37	drive t0 ...	9	7
38	drive t0 ...	10	7
39	load p5 ...	10	7
40	drive t0 ...	11	7
41	drive t0 ...	12	7
42	drive t0 ...	12	7
43	drive t0 ...	12	7
44	drive t0 ...	13	7
45	drive t0 ...	13	7

```
Steps:  19 /1575
Events: 105 /7771
Auto-steps per second (arrow up/down): 1
```

First time step where the heuristic value is reduced.

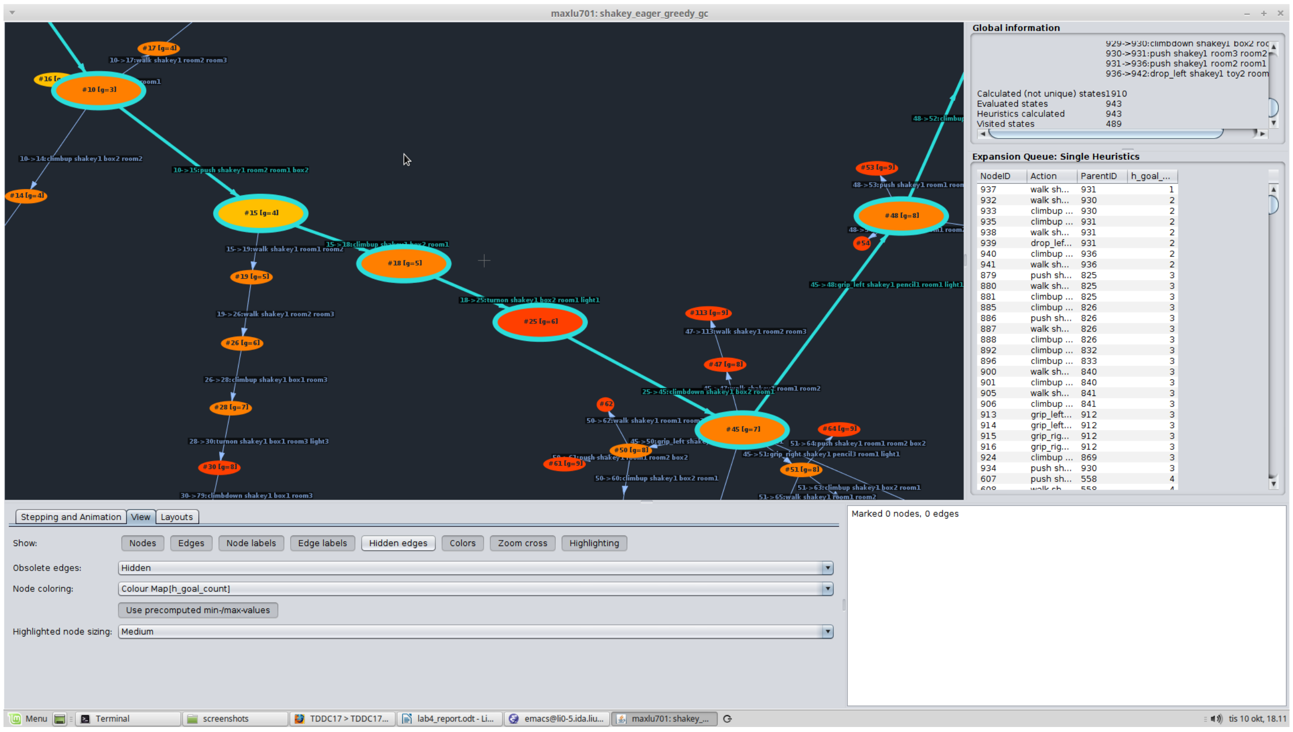
In the state that has progressed the longest at time step 27, the GC-heuristic still has 6 goals to expand.

The GC-heuristic never increase the heuristic value from one node to the next in the goal path.

3. Run one of the configurations on your own domain and problem. Is the graph similar to any of those that you have examined above? Can you explain why or why not?

Running our own problem in the visualiser, we used the goal-count heuristic. The results differed from the previous examples, in that we could see how the search expanded nodes that lowered the goal-count, but that did end up being dead-ends. In the previous examples, the lowering of the goal-count always seemed to get you closer to the goal-path, while in our example the goal-count even went down to 1 before going up to 5 again since the path did not end up being the correct goal-path. We can also see how the goal-path doesn't always decrease the goal-count for each node in the path, unlike the previous examples where every node on the goal-path had a lesser or equal goal-count as the previous node in the path.

The reason for this is perhaps that the goals are not in a sequential ordering, so that just clearing one goal is not guaranteed to give us a good path towards the final goal. For instance in our example with shakey, we want to have shakey end up in some specific room, but since the robot will have to move between rooms to clear all the goals, only going by a greedy heuristic that seeks to lower the goal-count will not guarantee to get us closer to the final goal.


maxlu701: shakey_eager_greedy_gc

Global information

```
                                  929->930:climbdown shakey1 box2 roo
                                  930->931:push shakey1 room3 room2
                                  931->936:push shakey1 room2 room1
                                  936->942:drop_left shakey1 toy2 room

Calculated (not unique) states1910
Evaluated states                  943
Heuristics calculated             943
Visited states                    489
```

Expansion Queue: Single Heuristics

NodeID	Action	ParentID	h_goal_...
937	walk sh...	931	1
932	walk sh...	930	2
933	climbup ...	930	2
935	climbup ...	931	2
938	walk sh...	931	2
939	drop_lef...	931	2
940	climbup ...	936	2
941	walk sh...	936	2
879	push sh...	825	3
880	walk sh...	825	3
881	climbup ...	825	3
885	climbup ...	826	3
886	push sh...	826	3
887	walk sh...	826	3
888	climbup ...	826	3
892	climbup ...	832	3
896	climbup ...	833	3
900	walk sh...	840	3
901	climbup ...	840	3
905	walk sh...	841	3
906	climbup ...	841	3
913	grip_left...	912	3
914	grip_left...	912	3
915	grip_rig...	912	3
916	grip_rig...	912	3
924	climbup ...	869	3
934	push sh...	930	3
607	push sh...	558	4
608	walk sh...	558	4

Stepping and Animation | View | Layouts

Show: Nodes, Edges, Node labels, Edge labels, Hidden edges, Colors, Zoom cross, Highlighting

Obsolete edges: Hidden

Node coloring: Colour Map[h_goal_count]

Use precomputed min-/max-values

Highlighted node sizing: Medium

Marked 0 nodes, 0 edges

The colors of the nodes in the goal-path represents the goal-counts. We can see that it goes from a lower goal-count (more yellow) to a higher goal-count (more red) in nodes that brings us closer to the goal-node.

Was any configuration better than the other? Was it better on everything or just on some problems?

The goal-count heuristic seems to work a lot better for the given examples, since it searched through a clear path to the goal without expanding a lot of dead-ends like the FF-heuristic did. However, we can see from the example with our own shakey-problem compared to the given examples that using the goal-count heuristic was a lot better in the given exampels than in the shakey problem. The reason for this is stated in the previous section.

Are your findings applicable to all the infinite many possible domains and problems? If so why? Else, why not?

Since we could already observe that one heuristic that seemed good for the example domain did not work at all work as well on our shakey problem, we can conclude that our findings are not applicable to every possible domain. The reason for the heuristic working so well in the given examples is probably because clearing one goal dependant on having cleared a previous goal, so it is not possible to clear a later goal before clearing a previous one. In our example with shakey, we instead might have to go back and forth between ”goals”, such as moving in and out of rooms, or turning on and off lights. Just trying to clear these goals without any regard to ordering of them will not get us closer to our final goal and therefore take us down dead-ends in our search.