List 3

Matheus Westhelle
Kayuã Oleques Paim

Exercise 1

(a) Provide planning tasks in positive normal form with the following characteristics, or justify why no such a task exists.

(i) A task Π_1 that is unsolvable, but with Π_1^+ having an optimal plan length of 2.

Let $\Pi_1 = \langle V, I, O, \gamma \rangle$, where $V = \{a, b, c\}$, $I = \{a\}$, $\gamma = a \wedge b \wedge c$, and we have the following operators O:

$$O_1 = \langle\ a, b, 1 \rangle$$
$$O_2 = \langle b, c \wedge \neg a, 1 \rangle$$

This task is unsolvable, as it's not possible to reach $a \wedge b \wedge c$ due to the $\neg a$ delete effect of operator O_2. We then define Π_1^+, which only differs from Π_1 on the O_2 operator:

$$O_2^+ = \langle b, c, 1 \rangle$$

For task Π_1^+, we can sequentially apply O_1^+ and O_2^+, which gives us a solution of length 2.

(ii) A task Π_2 with optimal plan length of 2, but such that Π_2^+ is unsolvable.

No such task exists. The relaxation lemma states that if a plan π leads to a goal state from state s, then π^+ leads to a goal state from s'. If there is no plan π^+ that leads to a goal state, as task Π_2^+ is unsolvable, by *modus tollens*, this violates the relaxation lemma.

(iii) A task Π_3 with set of operators $O = \{o_1, o_2, o_3\}$ such that Π_3^+ has an optimal plan of length 4.

Let $\Pi_3 = \langle V, I, O, \gamma \rangle$ with $V = \{r, s, t, u\}$, $I = \{r\}$, $\gamma = r \wedge s \wedge t \wedge u \wedge v$, and operators O:

$$O_1 = \langle r, s, 1 \rangle$$
$$O_2 = \langle s, t, 1 \rangle$$
$$O_3 = \langle t, u \wedge \neg t, 1 \rangle$$

Task Π_3 has the optimal plan $\pi^* = \langle O_1, O_2, O_3, O_2 \rangle$.

(iv) An infinite family of planning tasks $P = \{P_1, P_2, ...\}$ (e.g. the definition of P_i is parametrized by the value of integer parameter i) such that the optimal plan length of each task P_i increases with the value of i, but the optimal plan length of any P_i^+ is always 1.

Answer: Let a planning task P be defined parametrically as $V_i = \{v_1, v2..., v_{i1}\}$, $I_i = \{v_1\}$, $\gamma = v_1 \wedge ... \wedge v_i$. We then define, for each task P_i, operators as such:

$$O_1 = \langle\ v_1,\, v_2,\, 1\ \rangle$$
$$O_i = \langle\ v_{i-1},\, v_i,\, 1\ \rangle$$
$$\vdots$$
$$O_{i+1} = \langle\ \top,\, \neg\, v_1 \wedge v_2,\, ...,\, v_i,\, 1\ \rangle$$

The last operator has a delete effect for the variable in the initial state, but contains all other variables necessary for the goal, such that its relaxed counterpart O_{i+1}^+ is sufficient to form a plan by itself ($\pi_i = \langle\ O_{i+1}^+\ \rangle$) that reaches the goal from the initial state with length 1. For the original planning task, the optimal plan length grows as i increases.

(b) Take the simple instance of the Visitall domain (from the International Planning Competition) in directory visitall-untyped, and make sure you understand the problem. What is the optimal solution value $h^*(I)$?

For this instance of the problem, the robot starts in the midpoint of a single row of five rooms, and it must visit them all. For notation, we denote that the robot is at position i using r_i, and that position i has been visited with v_i. In this instance, the domain of i is 0, 1, 2, 3, 4. In order to visit all rooms, the robot has two optimal paths: it can either go all the way to the leftmost room r_0, and then "turn around" and move until the rightmost position, r_4; or it can start by moving all the way to the right, and then all the way to the left until r_0. The plans are:

$$\pi_1^* = \langle\ Move(r_2, r_1), Move(r_1, r_0), Move(r_0, r_1), Move(r_1, r_2), \\ Move(r_2, r_3), Move(r_3, r_4)\ \rangle$$

And

$$\pi_2^* = \langle\ Move(r_2, r_3), Move(r_3, r_4), Move(r_4, r_3), Move(r_3, r_2), \\ Move(r_2, r_1), Move(r_1, r_0)\ \rangle$$

Both plans have length 6. Thus, $h^*(I) = 6$.

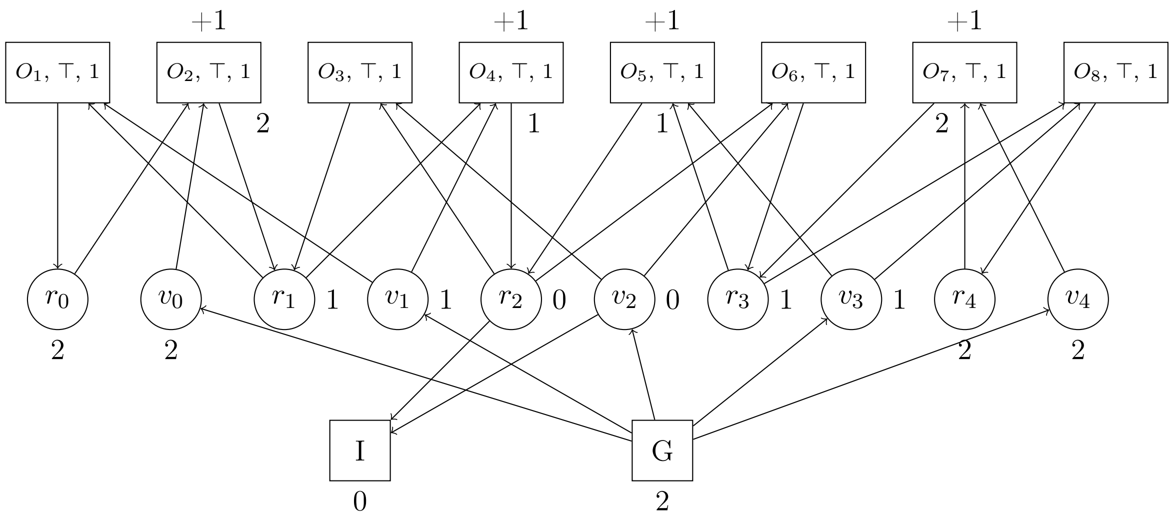
What is the value of $h^+(I)$?

For the relaxed version of the task, we don't account for the fact the robot needs to backtrack, as we don't consider the delete effects that remove him from the previous room at each movement. Thus, we simply visit one room at a time, as long as it is adjacent to rooms the robot has already visited. One optimal relaxed plan is:

$$\pi^+ = \langle\ Move(r_2, r_1),\ Move(r_1, r_0),\ Move(r_2, r_3),\ Move(r_3, r_4)\ \rangle$$

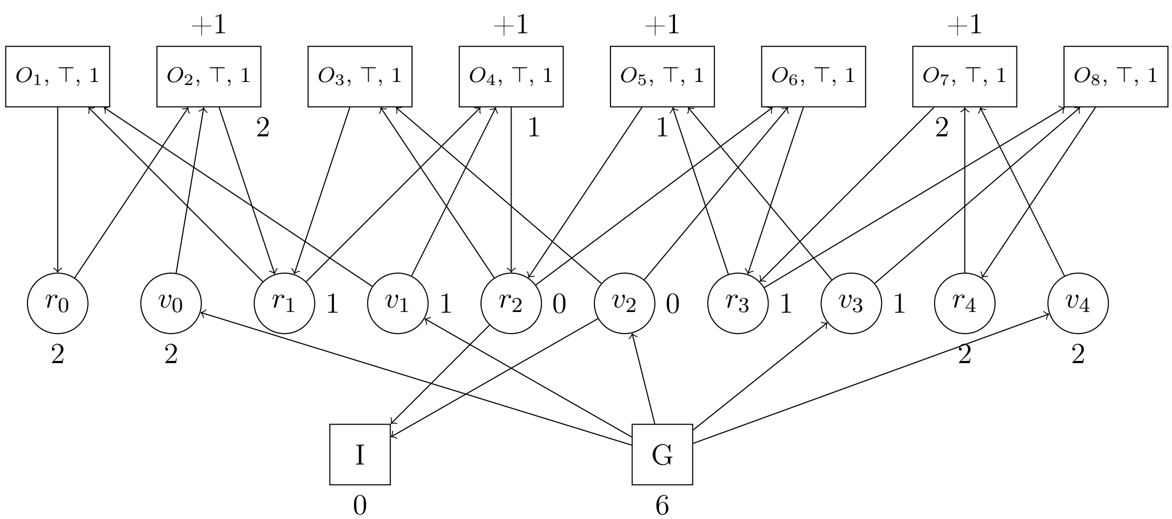
Thus, we have $h^+(I) = 4$.

Draw the full Relaxed Task Graph corresponding to the instance, and label each node with the final cost that results from (manually) applying the algorithm seen in class for computing h^{max}. What is the value of $h^{max}(I)$?



The value of $h^{max}(I)$ is 2.

Finally, label the graph again, but with the costs that result from h^{add} instead of h^{max}.



The value of $h^{add}(I)$ is 6.

	Number of Expansions		Time (s)	
Instance	Blind Search	Relaxed Task Graph	Blind Search	Relaxed Task Graph
castle 2-2-8	8	8	0.0014622	0.0013283
castle 3-3-8	84	84	0.0022348	0.0029757
castle 4-3-5	263	263	0.0034227	0.0071273
castle 5-4-7	12	6	0.003974	0.0049379
castle 5-4-9	7350	7350	0.02681	0.241813
castle 5-4-10	0	0	0.0009523	0.0009358
castle 6-4-7	24146	24022	0.070907	1.06481
castle 7-5-4	3680953	—	12.537	—
castle 8-5-9	4839625	—	16.9822	—
castle 9-6-5	661	206	0.0117677	0.0594118
castle 10-6-7	—	—	—	—
castle 12-7-3	4585	2520	0.0296206	0.810473
castle 16-9-1	22005	4382	0.101019	2.64418

Table 1: Comparison of blind search and A^* search on the relaxed task graph.

Exercise 2

For this exercise, we start by implementing an algorithm to compute the most conservative valuation of an AND/OR graph, and also implementing an algorithm that constructs a relaxed task graph for a STRIPS task. We then run the `fast-downward` planner using our heuristic and compare it against blind search by the number and speed of expansions on the *castle* domain.

As expected, the number of expansions of blind search is an upper bound on the number of expansions for the relaxed heuristic, as can be seen on table 1. However, computing the graph for the relaxed heuristic is generally costlier than blind search, which is reflected on the run times. This becomes more evident on harder instances such as `castle 7-5-4` and `castle 8-5-9`, where the computation blew past the time limit of 60 seconds when using the relaxed heuristic. These heuristics lack informedness, so neither are able to provide a full plan within the time limit for the `castle 10-6-7` instance.

Next, we implement the h^{add} heuristic, and use it to compare its initial heuristic values against the cost of an optimal relaxed plan, the actual discovered plan, and an optimal plan.

The comparison results can be seen on table 2 on the next page. It can be seen that the h^{add} heuristic severely overestimates many true costs of the relaxed tasks, from which it can be said that it is not an admissible heuristic. We also know that an optimal relaxed plan is never more expensive than an optimal plan for an original task, which is reflected on the comparison of the h^{+} and h^{*} columns.

The next experiment targets the h^{ff} heuristic. We implement it by computing a best achiever graph, and run the same experiment we did for h^{add}. The results can be seen on table 3 on the following page.

The initial state heuristic values for h^{ff} are all lower (often by a very large margin) than those for h^{add}, which happens because h^{ff} will not count multiple paths for the same operator as h^{add} does. Sometimes, such as for instance *castle 4-3-5*, h^{ff} will not find an optimal solution, but h^{add} will. This is likely due to the non-deterministic nature of h^{ff}. On most situations, however, h^{ff} finds better plans than h^{add}.

Instance	Initial h value (h^{add})	h^+	$\lvert\pi^{hadd}\rvert$	h^*
castle 2-2-8	6	4	4	4
castle 3-3-8	53	9	9	9
castle 4-3-5	161	14	14	15
castle 5-4-7	182	18*	∞	∞
castle 5-4-9	232	19	21	19
castle 5-4-10	∞	∞	∞	∞
castle 6-4-7	555	23*	26	26
castle 7-5-4	595	26*	40	35
castle 8-5-9	1704	28*	46	39
castle 9-6-5	2001	34*	∞	∞
castle 10-6-7	2257	36*	82	36*
castle 12-7-3	3348	42*	∞	∞
castle 16-9-1	10258	60*	∞	∞

Table 2: Comparison of plan lengths. **Actual value could not be computed, so we instead report the value of the last reached f-layer.*

Instance	Initial h value (h^{ff})	h^+	$\lvert\pi^{hff}\rvert$	h^*
castle 2-2-8	4	4	4	4
castle 3-3-8	9	9	9	9
castle 4-3-5	18	14	15	15
castle 5-4-7	25	18*	∞	∞
castle 5-4-9	25	19	19	19
castle 5-4-10	∞	∞	∞	∞
castle 6-4-7	34	23*	30	26
castle 7-5-4	41	26*	38	35
castle 8-5-9	47	28*	41	39
castle 9-6-5	56	34*	∞	∞
castle 10-6-7	61	36*	71	36*
castle 12-7-3	78	42*	∞	∞
castle 16-9-1	109	60*	∞	∞

Table 3: Comparison of plan lengths and performance of h^{ff}.