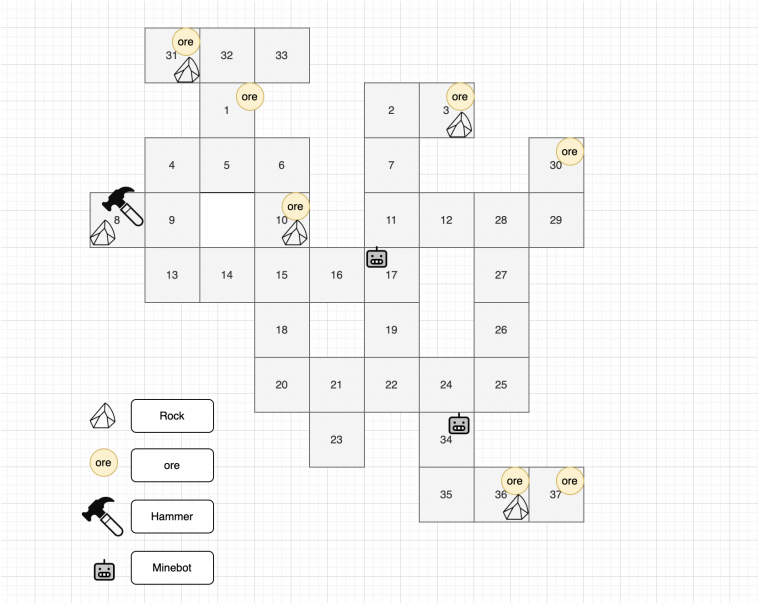
Informatics 2D Coursework 2 Report

1. (5 marks) Task 2.1 Design



This is the illustration of the new designed problem

In order to devise a harder problem for the ff planner a few given strategies were utilized and a harder problem called *problem − hard.pdll* was constructed as follows (Note : the major theme of each strategy involved increasing the search space) :

Additional MineBots : By assessing the domain file domain-1.pddl, it can clearly be seen that arguments of type robot is being passed onto every action. We already know that a planning problem is more challenging if it has a greater search space, which is the case when there are large number of types, predicates or possible actions present that can be executed. Thus it is a no-brainer to increase the number of robots as it exponentially increases the number of possible search states. This in turn increases the runtime for which the ff planner must run in order to provide a viable plan. Futhermore the planner must also consider which minebot would be best suited to get certain ores based on their location, in turn increases the number of possible states thus makes the problem more challenging. For my problem, I decided to add one extra robot and placed it at cell $c34$.

Additional Cells : In order to make the problem more difficult another thing that can be done is to increase the number of possible cells, I expanded mine to a total of 37 cells as arranged below. This increases the total number of possible paths that the minebot can take to reach a particular object or ore, thus increases the difficulty of the planning process as the planner has to now evaluate more possible states in order to find a solution. Once again the state space required to be searched to achieve the goal increases.

Increasing the number of Ores, rocks and altering the goal state : Finally in order to make the problem even more difficult for the planner, I increased the number of goal states which in this case were the number of ores to be mined and strategically blocked some of them like the ores at cell c10, c31 and c36, so that in some cases a blocked and unblocked ore are located nearby each other but far away from the initial position of the hammer. This way the planner has to make choices on

how the robot maneuvers and carries the hammer to break rocks in the cases where the ore is blocked. This condition makes the problem more complicated and requires the planner to search through a variety of states to find a solution, especially an optimal one, and allows the scope for improvement in the optimally department.

In order to illustrate the change in difficulty of the problem, two images have been attached below: The *first* showing the run time of the *problem* − 1.*pddl* and the *second* showing the run time of *problem* − *hard.pddl* with the run configuration: **./ff -o domain-1.pddl -f problem-hard.pddl -E**

```
       37: MOVE C18 C20 RM
       38: MOVE C20 C21 RM
       39: MOVE C21 LIFT RM
       40: LIFTON C RM
       41: MOVE LIFT C21 RM
       42: MOVE C21 C20 RM
       43: MOVE C20 C18 RM
       44: MOVE C18 C15 RM
       45: MOVE C15 C10 RM
       46: PICKUP C10 RM
       47: MOVE C10 C15 RM
       48: MOVE C15 C16 RM
       49: MOVE C16 C17 RM
       50: MOVE C17 C11 RM
       51: MOVE C11 C7 RM
       52: MOVE C7 C2 RM
       53: MOVE C2 C3 RM
       54: BREAKROCK C3 B RM
       55: DROPHOLDABLE C3 HAMMER RM
       56: GETORE C3 B RM
       57: MOVE C3 C2 RM
       58: MOVE C2 C7 RM
       59: MOVE C7 C11 RM
       60: MOVE C11 C17 RM
       61: MOVE C17 C19 RM
       62: MOVE C19 C22 RM
       63: MOVE C22 C21 RM
       64: MOVE C21 LIFT RM
       65: LIFTON B RM


time spent:    0.00 seconds instantiating 235 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 87 facts and 102 actions
               0.00 seconds creating final representation with 65 relevant facts, 0 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 184 states, to a max depth of 11
               0.00 seconds total time
```

```
      169: MOVE C31 C32 RM
      170: MOVE C32 C1 RM
      171: MOVE C1 C5 RM
      172: MOVE C5 C6 RM
      173: MOVE C6 C10 RM
      174: MOVE C10 C15 RM
      175: MOVE C15 C18 RM
      176: MOVE C18 C20 RM
      177: MOVE C20 C21 RM
      178: MOVE C21 LIFT RM
      179: LIFTON F RM
      180: MOVE LIFT C21 RM
      181: MOVE C21 C20 RM
      182: MOVE C20 C18 RM
      183: MOVE C18 C15 RM
      184: MOVE C15 C10 RM
      185: MOVE C10 C6 RM
      186: MOVE C6 C5 RM
      187: MOVE C5 C1 RM
      188: GETORE C1 A RM
      189: MOVE C1 C5 RM
      190: MOVE C5 C6 RM
      191: MOVE C6 C10 RM
      192: MOVE C10 C15 RM
      193: MOVE C15 C18 RM
      194: MOVE C18 C20 RM
      195: MOVE C20 C21 RM
      196: MOVE C21 LIFT RM
      197: LIFTON A RM


time spent:    0.00 seconds instantiating 1354 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 194 facts and 340 actions
               0.00 seconds creating final representation with 160 relevant facts, 0 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               2.77 seconds searching, evaluating 48275 states, to a max depth of 0
               2.77 seconds total time
```

As seen from the images above, the problem has clearly been made more challenging as now 48275 states are being evaluated as compared to 339 previously, hence with the new configuration the planner is taking more time to devise a legal plan under best first search.

2. (10 marks) Task 2.2 Evaluation

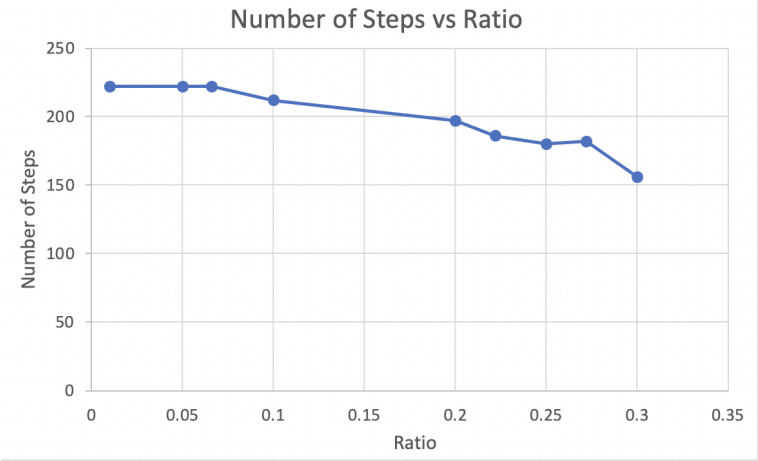
The aim of this task is to design an experiment to evaluate the effect of different values of w_g and w_h on the planner's performance under the $best - first$ search configuration. Thus in order to do some experimentation, the planner was first ran on $problem - hard.pdll$ using random w_g and w_h values on doing so a key observation was made that the planner's performance somehow related to the ration between w_g and w_h. In order to explore this the following experiment was devised:

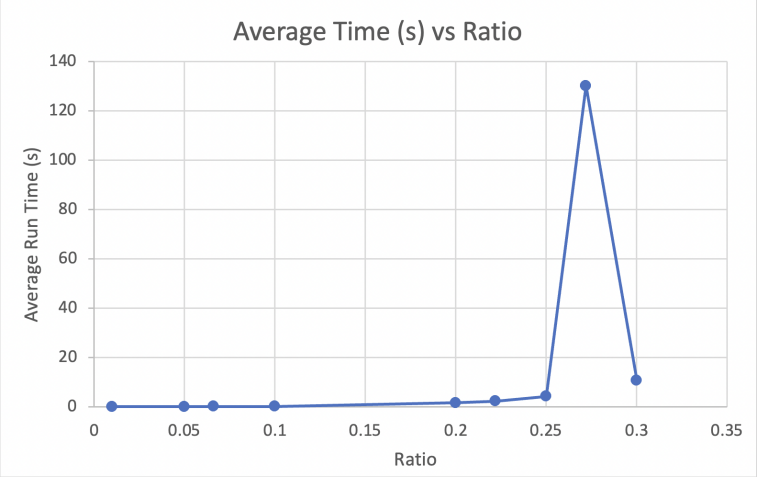
- *Values of w_g and w_h were selected such that the ratio w_g/w_h varied from 0 and moved towards 1.*
- *Values of the number of steps and the time taken for each run was noted, in fact each configuration of w_h and w_g was run three times and its average run time was computed and compared with the rest of the runs.*
- *The observations from the experiment will then be tabulated and graphed, in order to analyse and draw conclusions.*

The experiment values after running the *problem* through the panner with various w_g and w_h configurations are :

W_g	W_h	Ratio	Time1 (S)	Time2 (s)	Time3 (s)	Average Time (s)	Number of Steps
1	100	0.01	0.13	0.11	0.11	0.1166	222
1	50	0.05	0.12	0.13	0.13	0.1266	222
3	45	0.066	0.14	0.14	0.13	0.205	222
1	10	0.1	0.25	0.26	0.26	0.256	212
2	10	0.1	0.25	0.27	0.26	0.26	212
2	10	0.2	1.68	1.75	1.66	1.696	197
2	9	0.222	2.25	2.32	2.58	2.383	186
1	4	0.25	4.17	4.45	4.32	4.313	180
3	11	0.272	129.19	129.77	130	129.65	182
3	10	0.3	11.2	10.58	10.63	10.8	156
1	3	0.33	-	-	-	Too long	-

Using the above table the following graphs can be generates to visualise the data:





Earlier we had mentioned that the planner's performance under the $best - first$ search configuration was somehow related to the ratio of w_g and w_h, this can be formally verified as the original paper on the FF planner uses w_g and w_h as heuristic weight values that balance the heuristic and the goal distance components in the computation of the heuristic value for best-first search. That is w_g value is the weight given to the heuristic estimate of the remaining cost to reach the goal state from the current state. A higher w_g value will result in a more greedy search, focusing more on the heuristic estimate and less on the actual cost of reaching the goal state, where as the w_h value is the weight given to the actual cost of reaching the current state from the start state. A higher w_h value will result in a more conservative search, considering the actual cost of reaching the current state more strongly. Thus the ratio of these 2 components determines how much of the heuristic is based on the remaining cost to reach the goal state from the current state and how much of it is based on the cost of reaching the current state from the goal state.

Referring to the figures above for the Number of Steps vs ratio and Average Time vs ratio, and by other random experiments, the following conclusions or observations can be drawn:

1. In general for lower ratios of $w_g : w_h$ the search was less optimal and the planner took less time to figure our a legal plan as compared to the higher ratios where the planner seemed to have become more optimal but the running time significantly increased.
2. The running time of the algorithm increased as we moved towards a greater ratio, this basically meant that the search became more greedy, and as a result the number of states the planner had to evaluate increased as many states may have had similar heuristic function evaluations. As for this problem, it is easier to rank based on the distance from the start to the current state vs the distance from the current to the goal state, since many states have similar distances to the goal state and it isnt clear how much this might be, as evaluating this requires a lot of work. Hence the long running times.
3. In this case, and I believe this can be extended to others as well, the optimally only started increasing after a certain ratio (0.1), and all other ratios like 0.01 vs 0.066 gave the same number of steps even though it took almost twice the time to run for the higher ratio (0.1 sec more).
4. The optimality increased or the number of steps in the solution decreased as the ratio increased, as it can be seen from the graph. However the experiment only continued up until a ratio of 0.3, above that all other configurations took too long to complete and was thus not included in the analysis. A fair guess would be that the most optimal results are obtained by taking more or less around $1 : 1$ ratio between the 2 weights, as the costs of reaching the current state from the start and start to goal are both equally important in most cases. However this would require tremendous running time and in most use cases this ratio must be experimented with in different planning problems to find the ratio that provides the maximum optimality.
5. From the previous point, around the ratio 0.3 was when the ff planner was producing an optimal solution in decent time (10 seconds) for this problem, even though for some ratios like 0.272 which

is lower than 0.3, it still ran for about 2 minutes and failed to produce a very optimal solution. Thus it really shows that experimenting with various w_g and w_h values are required regardless of the trends in order to find the best ratio suitable for providing an optimal enough solution in an acceptable time. However we can use these analysis to make educated guesses about the w_h and w_g values which will speed up the experimentation process greatly.

3. (25 marks) Task 3.4 Your Extension

Your answer here.