Simulation of a Search and Rescue Robot: Problem Analysis

CW REPORT FOR 4CCS1IAI INTRODUCTION TO ARTIFICIAL INTELLIGENCE

By

Fernando Lee | 1631508
Christian Honkanen | 1631080
Vishnu Thirumalai | 1631454
Sakeen Zaman | 1631323

Analysis of Problem Files

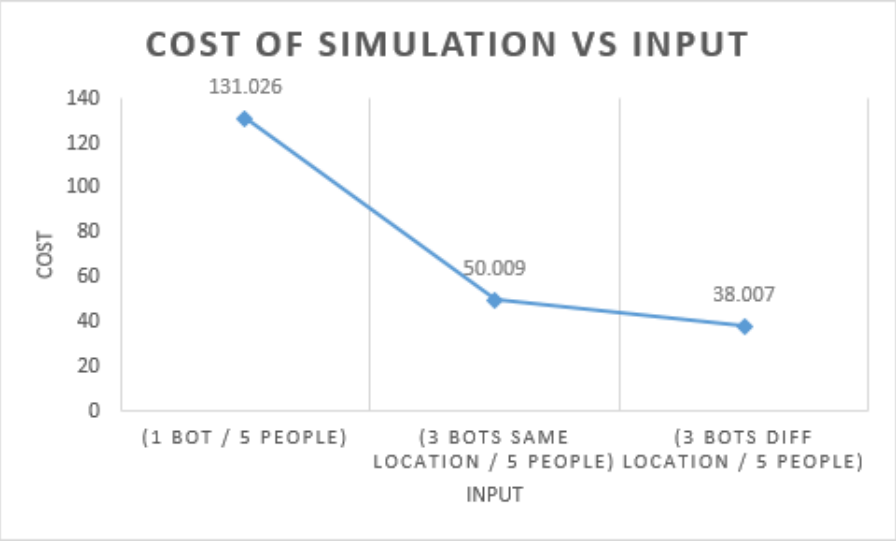
To accurately see how the planner dealt with locating the victims in the quickest times possible, three scenarios were considered as outlined before. For a closer look at the problem files, please see Appendices 2.1, 2.2 and 2.3

For these problems, the same 5x5 grid was used throughout, for a more accurate comparison. Appendices 4.1, 4.2 and 4.3 contain the diagrams for each problem, which only vary the number and positions of the robots. The timings for each action remain the same.

Appendices 4.1, 4.2 and 4.3 also show the most optimised route the planner was able to come up with for each problem. Appendices 3.1, 3.2 and 3.3 show the plan for each of these routes.

A planner could potentially further optimise these plans; however OPTIC was unable to do so due to exponential increases in the number of states considered. When it hits about 461,000 states it runs out of memory, throws a “std :bad_alloc” and creates a stackdump. Initially a 6x6 grid was used, but the problems were simplified it to 5x5 grid and multiple obstacles were removed so OPTIC would be able to optimise to a reasonable level.

Costs of Simulations vs. Input Sizes

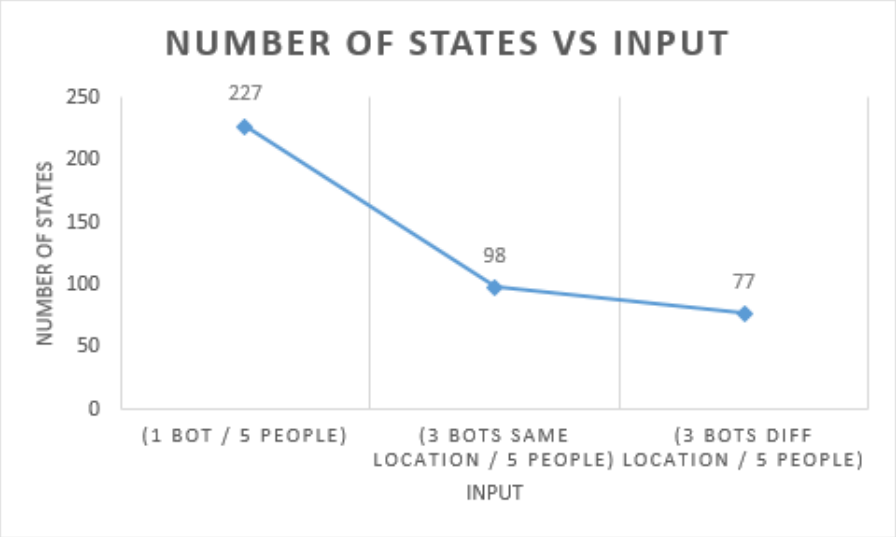


In the graph above, the time taken to locate all victims from the three scenarios has been plotted against the number of robots and their starting locations.

The graph clearly shows that the best option would be to have multiple robots entering the building from different locations as there is almost a 12 second improvement from having them start in the same location. This is because they have a wider initial reach and the robots will focus on the victims closest to them. This can be seen in Appendix 4.1, 4.2 and 4.3 where the actions of the robots have been mapped.

Having one robot will takes far too much time as it needs to have the single robot visit all of the victims, rather than splitting the visits between the robots. This can be seen in the graph as a single robot takes almost 162% more time to locate all the victims when compared to the having 3 robots. This however assumes the single robot has the same speed as the 3 robots. Further tests where the single robot was faster than the 3 robots lowered this margin, but not by enough to justify a single robot based on speed alone.

Number of States vs. Input Sizes



The above graph shows that the planner creates more states when a single robot is used, even though multiple robots would create more choices at each level. This is most likely due to a single robot simply having far more levels of choice than multiple robots. i.e since it takes longer to find the people because it's performing all the actions itself.

This data directly agrees with the cost vs. input graph, with similar drops (61% drop vs. 56% drop for 1 robot to 3 robots, and a 34% vs 32% drop for the same location against multiple locations) in y-axis values.

Time Taken vs. Number of states evaluated

3 robots starting from multiple locations:

Time taken (s)	Cost	States evaluated	Error?
0.53	38.007	77	
0.74	30.005	333	
???	???	???	Stackdump occurs, out of memory

3 robots starting from the same location:

Time taken (s)	Cost	States evaluated	Error?
0.62	50.009	98	
1.64	46.009	850	
24.67	42.007	15257	
???	???	???	Stackdump occurs, out of memory

1 robot:

Time taken (s)	Cost	States evaluated	Error?
0.6	131.026	227	
3.7	129.026	3161	
6.7	118.025	6182	
6.81	109.023	6273	
13.96	107.021	13978	
14.28	100.021	14314	
14.46	96.021	14524	
30.26	89.018	32545	
39.07	85.018	42119	
???	???	???	Stackdump occurs, out of memory

From the tables, we can see that the single bot problem can be optimised many more times before the program runs out of memory. We also see that at the end of the simulation, the program has evaluated many more states compared to the other two. We can also conclude that even though the first simulation does get to optimise the solution many more times than the last two, it still results in a stackdump at the end. This most likely means that the computers that the simulations were run on couldn't handle the amount of actions being performed, or that there are just too many things that the program is doing, and ends up terminating regardless.

As the time taken to get to a more optimised solution (lower cost) increases, so does the number of states evaluated at the end. This is the most likely reason that the stackdump is caused at the end, and we therefore have much less optimised solutions for the 3 robot problems compared to the 1 robot problem. It also seems like the states for 1 robot are "simpler" than the other two. This can be based off of the fact that after 39 seconds in problem 1, we have 42000 states evaluated. In problem 2, after 24.7 seconds, we have only 15000 in comparison. Comparing those 2 values shows that even though the time taken is about 1.6 times longer, the number of states is almost 3 times as much, or 2.8 to be exact, hence the program goes through more much faster.

Appendix 2.1: Problem file 1 (one robot)

```
(define (problem DisasterRobotProblem1)
	(:domain DisasterRobot)

	(:objects
	 Robo1 - robot
	 p1 p2 p3 p4 p5 - person
	 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16 a17 a18 a19 a20 a21 a22 a23 a24 a25 -
location
	 Puddle1 Puddle2 Pillar1 Pillar2 Wall1 Wall2 Wall3 Rubble1 Rubble2 Rubble3 - obstacle
	)

	(:init
		(at Robo1 a1)
		(notBusy Robo1)

		(at p1 a4)
		(clear a4)
		(noInjection p1)
		(needsMark p1)
		(notBeingInteractedWith p1)

		(at p2 a11)
		(clear a11)
		(needsMark p2)
		(notBeingInteractedWith p2)

		(at p3 a18)
		(clear a18)
		(noInjection p3)
		(needsMark p3)
		(notBeingInteractedWith p3)

		(at p4 a22)
		(clear a22)
		(needsMark p4)
		(notBeingInteractedWith p4)

		 (at p5 a15)
		(clear a15)
		(noInjection p5)
		(needsMark p5)
		(notBeingInteractedWith p5)

		(at Puddle1 a5)
		(canGoThrough Puddle1)
		(notBeingInteractedWith Puddle1)

		(at Puddle2 a12)
		(canGoThrough Puddle2)
		(notBeingInteractedWith Puddle2)

		(at Wall1 a6)
		(notLoadBearing Wall1)
		(canDrillThrough Wall1)
		(notBeingInteractedWith Wall1)

		(at Wall2 a9)
		(notLoadBearing Wall2)
		(canDrillThrough Wall2)
		(notBeingInteractedWith Wall2)

		(at Wall3 a23)
```

(notLoadBearing Wall3)
(canDrillThrough Wall3)
(notBeingInteractedWith Wall3)

(at Pillar1 a7)
(canGoAround Pillar1)
(notBeingInteractedWith Pillar1)

(at Pillar2 a19)
(canGoAround Pillar2)
(notBeingInteractedWith Pillar2)

(at Rubble1 a3)
(notLoadBearing Rubble1)
(canBePushed Rubble1)
(notBeingInteractedWith Rubble1)

(at Rubble2 a17)
(notLoadBearing Rubble2)
(canBePushed Rubble2)
(notBeingInteractedWith Rubble2)

(at Rubble3 a14)
(notLoadBearing Rubble3)
(canGoOver Rubble3)
(notBeingInteractedWith Rubble3)

(clear a1)
(adjacent a1 a2)
(adjacent a1 a6)

(clear a2)
(adjacent a2 a1)
(adjacent a2 a3)
(adjacent a2 a7)

(not(clear a3))
(adjacent a3 a2)
(adjacent a3 a4)
(adjacent a3 a8)

(clear a4)
(adjacent a4 a3)
(adjacent a4 a5)
(adjacent a4 a9)

(not(clear a5))
(adjacent a5 a4)
(adjacent a5 a10)

(not(clear a6))
(adjacent a6 a7)
(adjacent a6 a1)
(adjacent a6 a11)

(not(clear a7))
(adjacent a7 a6)
(adjacent a7 a8)
(adjacent a7 a2)
(adjacent a7 a12)

(clear a8)
(adjacent a8 a7)
(adjacent a8 a9)
(adjacent a8 a3)
(adjacent a8 a13)

(not(clear a9))
(adjacent a9 a8)
(adjacent a9 a10)
(adjacent a9 a4)
(adjacent a9 a14)

(clear a10)
(adjacent a10 a9)
(adjacent a10 a5)
(adjacent a10 a15)

(clear a11)
(adjacent a11 a12)
(adjacent a11 a6)
(adjacent a11 a16)

(not(clear a12))
(adjacent a12 a11)
(adjacent a12 a13)
(adjacent a12 a7)
(adjacent a12 a17)

(clear a13)
(adjacent a13 a12)
(adjacent a13 a14)
(adjacent a13 a8)
(adjacent a13 a18)

(not(clear a14))
(adjacent a14 a13)
(adjacent a14 a15)
(adjacent a14 a9)
(adjacent a14 a19)

(clear a15)
(adjacent a15 a14)
(adjacent a15 a10)
(adjacent a15 a20)

(clear a16)
(adjacent a16 a17)
(adjacent a16 a11)
(adjacent a16 a21)

(not(clear a17))
(adjacent a17 a16)
(adjacent a17 a18)
(adjacent a17 a12)
(adjacent a17 a22)

(clear a18)
(adjacent a18 a17)
(adjacent a18 a19)
(adjacent a18 a13)
(adjacent a18 a23)

(not(clear a19))
(adjacent a19 a18)
(adjacent a19 a20)
(adjacent a19 a14)
(adjacent a19 a24)

(clear a20)
(adjacent a20 a19)
(adjacent a20 a15)
(adjacent a20 a25)

```
        (clear a21)
        (adjacent a21 a22)
        (adjacent a21 a16)

        (clear a22)
        (adjacent a22 a21)
        (adjacent a22 a23)
        (adjacent a22 a17)

        (not(clear a23))
        (adjacent a23 a22)
        (adjacent a23 a24)
        (adjacent a23 a18)

        (clear a24)
        (adjacent a24 a23)
        (adjacent a24 a25)
        (adjacent a24 a19)

        (clear a25)
        (adjacent a25 a24)
        (adjacent a25 a20)

        (= (normalMove Robo1) 4)
        (= (injectionTime) 6)
        (= (markTime) 2)

        (= (goIntoTime Puddle1) 3)
        (= (goIntoTime Puddle2) 5)

        (= (goAroundTime Pillar1) 6)
        (= (goAroundTime Pillar2) 8)

        (= (drillTime Wall1) 10)
        (= (drillTime Wall3) 10)
        (= (drillTime Wall2) 12)

        (= (pushTime Rubble1) 6)
        (= (pushTime Rubble2) 6)
        (= (pushTime Rubble3) 8)
    )

    (:goal (and (marked p1)
            (marked p2)
            (marked p3)
            (marked p4)
            (marked p5))
    )
)
```

Appendix 2.2: Problem file 2 (three robots in different start locations)

```
(define (problem DisasterRobotProblem2)
    (:domain DisasterRobot)

    (:objects
     Robo1 Robo2 Robo3 - robot
     p1 p2 p3 p4 p5 - person
     a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16 a17 a18 a19 a20 a21 a22 a23 a24 a25 -
location
     Puddle1 Puddle2 Pillar1 Pillar2 Wall1 Wall2 Wall3 Rubble1 Rubble2 Rubble3 - obstacle
    )

    (:init
```

(at Robo1 a1)
(notBusy Robo1)

(at Robo2 a10)
(notBusy Robo2)

(at Robo3 a24)
(notBusy Robo3)

(at p1 a4)
(clear a4)
(noInjection p1)
(needsMark p1)
(notBeingInteractedWith p1)

(at p2 a11)
(clear a11)
(needsMark p2)
(notBeingInteractedWith p2)

(at p3 a18)
(clear a18)
(noInjection p3)
(needsMark p3)
(notBeingInteractedWith p3)

(at p4 a22)
(clear a22)
(needsMark p4)
(notBeingInteractedWith p4)

(at p5 a15)
(clear a15)
(noInjection p5)
(needsMark p5)
(notBeingInteractedWith p5)

(at Puddle1 a5)
(canGoThrough Puddle1)
(notBeingInteractedWith Puddle1)

(at Puddle2 a12)
(canGoThrough Puddle2)
(notBeingInteractedWith Puddle2)

(at Wall1 a6)
(notLoadBearing Wall1)
(canDrillThrough Wall1)
(notBeingInteractedWith Wall1)

(at Wall2 a9)
(notLoadBearing Wall2)
(canDrillThrough Wall2)
(notBeingInteractedWith Wall2)

(at Wall3 a23)
(notLoadBearing Wall3)
(canDrillThrough Wall3)
(notBeingInteractedWith Wall3)

(at Pillar1 a7)
(canGoAround Pillar1)
(notBeingInteractedWith Pillar1)

(at Pillar2 a19)
(canGoAround Pillar2)
(notBeingInteractedWith Pillar2)

(at Rubble1 a3)
(notLoadBearing Rubble1)
(canBePushed Rubble1)
(notBeingInteractedWith Rubble1)

(at Rubble2 a17)
(notLoadBearing Rubble2)
(canBePushed Rubble2)
(notBeingInteractedWith Rubble2)

(at Rubble3 a14)
(notLoadBearing Rubble3)
(canGoOver Rubble3)
(notBeingInteractedWith Rubble3)

(clear a1)
(adjacent a1 a2)
(adjacent a1 a6)

(clear a2)
(adjacent a2 a1)
(adjacent a2 a3)
(adjacent a2 a7)

(not(clear a3))
(adjacent a3 a2)
(adjacent a3 a4)
(adjacent a3 a8)

(clear a4)
(adjacent a4 a3)
(adjacent a4 a5)
(adjacent a4 a9)

(not(clear a5))
(adjacent a5 a4)
(adjacent a5 a10)

(not(clear a6))
(adjacent a6 a7)
(adjacent a6 a1)
(adjacent a6 a11)

(not(clear a7))
(adjacent a7 a6)
(adjacent a7 a8)
(adjacent a7 a2)
(adjacent a7 a12)

(clear a8)
(adjacent a8 a7)
(adjacent a8 a9)
(adjacent a8 a3)
(adjacent a8 a13)

(not(clear a9))
(adjacent a9 a8)
(adjacent a9 a10)
(adjacent a9 a4)
(adjacent a9 a14)

(clear a10)
(adjacent a10 a9)
(adjacent a10 a5)
(adjacent a10 a15)

(clear a11)
(adjacent a11 a12)
(adjacent a11 a6)
(adjacent a11 a16)

(not(clear a12))
(adjacent a12 a11)
(adjacent a12 a13)
(adjacent a12 a7)
(adjacent a12 a17)

(clear a13)
(adjacent a13 a12)
(adjacent a13 a14)
(adjacent a13 a8)
(adjacent a13 a18)

(not(clear a14))
(adjacent a14 a13)
(adjacent a14 a15)
(adjacent a14 a9)
(adjacent a14 a19)

(clear a15)
(adjacent a15 a14)
(adjacent a15 a10)
(adjacent a15 a20)

(clear a16)
(adjacent a16 a17)
(adjacent a16 a11)
(adjacent a16 a21)

(not(clear a17))
(adjacent a17 a16)
(adjacent a17 a18)
(adjacent a17 a12)
(adjacent a17 a22)

(clear a18)
(adjacent a18 a17)
(adjacent a18 a19)
(adjacent a18 a13)
(adjacent a18 a23)

(not(clear a19))
(adjacent a19 a18)
(adjacent a19 a20)
(adjacent a19 a14)
(adjacent a19 a24)

(clear a20)
(adjacent a20 a19)
(adjacent a20 a15)
(adjacent a20 a25)

(clear a21)
(adjacent a21 a22)
(adjacent a21 a16)

(clear a22)
(adjacent a22 a21)
(adjacent a22 a23)
(adjacent a22 a17)

(not(clear a23))
(adjacent a23 a22)

```
        (adjacent a23 a24)
        (adjacent a23 a18)

        (clear a24)
        (adjacent a24 a23)
        (adjacent a24 a25)
        (adjacent a24 a19)

        (clear a25)
        (adjacent a25 a24)
        (adjacent a25 a20)


        (= (normalMove Robo1) 4)
        (= (injectionTime) 6)
        (= (markTime) 2)

        (= (normalMove Robo2) 4)
        (= (injectionTime) 6)
        (= (markTime) 2)

        (= (normalMove Robo3) 4)
        (= (injectionTime) 6)
        (= (markTime) 2)

        (= (goIntoTime Puddle1) 3)
        (= (goIntoTime Puddle2) 5)

        (= (goAroundTime Pillar1) 6)
        (= (goAroundTime Pillar2) 8)

        (= (drillTime Wall1) 10)
        (= (drillTime Wall3) 10)
        (= (drillTime Wall2) 12)

        (= (pushTime Rubble1) 6)
        (= (pushTime Rubble2) 6)
        (= (pushTime Rubble3) 8)

    )

    (:goal (and (marked p1)
            (marked p2)
            (marked p3)
            (marked p4)
            (marked p5))
    )
)
```

Appendix 2.3: Problem file 3 (three robots starting from same location)

```
(define (problem DisasterRobotProblem2)
    (:domain DisasterRobot)

    (:objects
     Robo1 Robo2 Robo3 - robot
     p1 p2 p3 p4 p5 - person
     a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16 a17 a18 a19 a20 a21 a22 a23 a24 a25 -
location
     Puddle1 Puddle2 Pillar1 Pillar2 Wall1 Wall2 Wall3 Rubble1 Rubble2 Rubble3 - obstacle
```

```
)

(:init
    (at Robo1 a1)
    (notBusy Robo1)

    (at Robo2 a1)
    (notBusy Robo2)

    (at Robo3 a1)
    (notBusy Robo3)

    (at p1 a4)
    (clear a4)
    (noInjection p1)
    (needsMark p1)
    (notBeingInteractedWith p1)

    (at p2 a11)
    (clear a11)
    (needsMark p2)
    (notBeingInteractedWith p2)

    (at p3 a18)
    (clear a18)
    (noInjection p3)
    (needsMark p3)
    (notBeingInteractedWith p3)

    (at p4 a22)
    (clear a22)
    (needsMark p4)
    (notBeingInteractedWith p4)

    (at p5 a15)
    (clear a15)
    (noInjection p5)
    (needsMark p5)
    (notBeingInteractedWith p5)

    (at Puddle1 a5)
    (canGoThrough Puddle1)
    (notBeingInteractedWith Puddle1)

    (at Puddle2 a12)
    (canGoThrough Puddle2)
    (notBeingInteractedWith Puddle2)

    (at Wall1 a6)
    (notLoadBearing Wall1)
    (canDrillThrough Wall1)
    (notBeingInteractedWith Wall1)

    (at Wall2 a9)
    (notLoadBearing Wall2)
    (canDrillThrough Wall2)
    (notBeingInteractedWith Wall2)

    (at Wall3 a23)
    (notLoadBearing Wall3)
    (canDrillThrough Wall3)
    (notBeingInteractedWith Wall3)

    (at Pillar1 a7)
    (canGoAround Pillar1)
    (notBeingInteractedWith Pillar1)
```

(at Pillar2 a19)
(canGoAround Pillar2)
(notBeingInteractedWith Pillar2)

(at Rubble1 a3)
(notLoadBearing Rubble1)
(canBePushed Rubble1)
(notBeingInteractedWith Rubble1)

(at Rubble2 a17)
(notLoadBearing Rubble1)
(canBePushed Rubble2)
(notBeingInteractedWith Rubble2)

(at Rubble3 a14)
(notLoadBearing Rubble1)
(canGoOver Rubble3)
(notBeingInteractedWith Rubble3)

(clear a1)
(adjacent a1 a2)
(adjacent a1 a6)

(clear a2)
(adjacent a2 a1)
(adjacent a2 a3)
(adjacent a2 a7)

(not(clear a3))
(adjacent a3 a2)
(adjacent a3 a4)
(adjacent a3 a8)

(clear a4)
(adjacent a4 a3)
(adjacent a4 a5)
(adjacent a4 a9)

(not(clear a5))
(adjacent a5 a4)
(adjacent a5 a10)

(not(clear a6))
(adjacent a6 a7)
(adjacent a6 a1)
(adjacent a6 a11)

(not(clear a7))
(adjacent a7 a6)
(adjacent a7 a8)
(adjacent a7 a2)
(adjacent a7 a12)

(clear a8)
(adjacent a8 a7)
(adjacent a8 a9)
(adjacent a8 a3)
(adjacent a8 a13)

(not(clear a9))
(adjacent a9 a8)
(adjacent a9 a10)
(adjacent a9 a4)
(adjacent a9 a14)

(clear a10)
(adjacent a10 a9)

(adjacent a10 a5)
(adjacent a10 a15)

(clear a11)
(adjacent a11 a12)
(adjacent a11 a6)
(adjacent a11 a16)

(not(clear a12))
(adjacent a12 a11)
(adjacent a12 a13)
(adjacent a12 a7)
(adjacent a12 a17)

(clear a13)
(adjacent a13 a12)
(adjacent a13 a14)
(adjacent a13 a8)
(adjacent a13 a18)

(not(clear a14))
(adjacent a14 a13)
(adjacent a14 a15)
(adjacent a14 a9)
(adjacent a14 a19)

(clear a15)
(adjacent a15 a14)
(adjacent a15 a10)
(adjacent a15 a20)

(clear a16)
(adjacent a16 a17)
(adjacent a16 a11)
(adjacent a16 a21)

(not(clear a17))
(adjacent a17 a16)
(adjacent a17 a18)
(adjacent a17 a12)
(adjacent a17 a22)

(clear a18)
(adjacent a18 a17)
(adjacent a18 a19)
(adjacent a18 a13)
(adjacent a18 a23)

(not(clear a19))
(adjacent a19 a18)
(adjacent a19 a20)
(adjacent a19 a14)
(adjacent a19 a24)

(clear a20)
(adjacent a20 a19)
(adjacent a20 a15)
(adjacent a20 a25)

(clear a21)
(adjacent a21 a22)
(adjacent a21 a16)

(clear a22)
(adjacent a22 a21)
(adjacent a22 a23)
(adjacent a22 a17)

```
        (not(clear a23))
        (adjacent a23 a22)
        (adjacent a23 a24)
        (adjacent a23 a18)

        (clear a24)
        (adjacent a24 a23)
        (adjacent a24 a25)
        (adjacent a24 a19)

        (clear a25)
        (adjacent a25 a24)
        (adjacent a25 a20)


        (= (normalMove Robo1) 4)
        (= (normalMove Robo2) 4)
        (= (normalMove Robo3) 4)
        (= (injectionTime) 6)
        (= (markTime) 2)

        (= (goIntoTime Puddle1) 3)
        (= (goIntoTime Puddle2) 5)

        (= (goAroundTime Pillar1) 6)
        (= (goAroundTime Pillar2) 8)

        (= (drillTime Wall1) 10)
        (= (drillTime Wall3) 10)
        (= (drillTime Wall2) 12)

        (= (pushTime Rubble1) 6)
        (= (pushTime Rubble2) 6)
        (= (pushTime Rubble3) 8)
    )

    (:goal (and (marked p1)
            (marked p2)
            (marked p3)
            (marked p4)
            (marked p5))
    )
)
```

Appendix 3.1: OUTPUT of Problem file 1 (one robot)

```
Initial heuristic = 40.000, admissible cost estimate 0.000
b (39.000 | 10.000)b (36.000 | 14.000)b (35.000 | 14.001)b (34.000 | 20.002)b (32.000 | 26.003)
Resorting to best-first search
Running WA* with W = 5.000, not restarting with goal states
b (39.000 | 10.000)b (36.000 | 14.000)b (35.000 | 14.001)b (34.000 | 20.002)b (34.000 | 18.002)b (32.000 | 26.003)b (31.000 | 24.003)b (30.
000 | 36.005)b (27.000 | 42.006)b (26.000 | 48.008)b (25.000 | 52.008)b (24.000 | 64.010)b (24.000 | 62.010)b (23.000 | 62.011)b (22.000 |
66.011)b (19.000 | 71.013)b (18.000 | 77.013)b (18.000 | 75.013)b (16.000 | 86.016)b (15.000 | 86.016)b (14.000 | 92.017)b (14.000 | 90.017
)b (13.000 | 100.019)b (12.000 | 104.020)b (11.000 | 104.020)b (10.000 | 109.021)b (9.000 | 109.021)b (8.000 | 115.022)b (7.000 | 115.022)b
 (6.000 | 121.023)b (5.000 | 125.024)b (4.000 | 125.024)b (3.000 | 129.025)b (2.000 | 129.025)b (1.000 | 131.026)(G);;;; Solution Found
; States evaluated: 227
; Cost: 131.026
; Time 0.70
0.000: (drill a1 a6 robo1 wall1)  [10.000]
10.001: (move a1 a6 robo1)  [4.000]
14.002: (move-around a6 a7 a8 robo1 pillar1)  [6.000]
20.003: (move a8 a13 robo1)  [4.000]
24.004: (move a13 a18 robo1)  [4.000]
28.005: (push-aside a18 a17 robo1 rubble2)  [6.000]
34.006: (move-around a18 a19 a20 robo1 pillar2)  [8.000]
42.007: (move a20 a15 robo1)  [4.000]
46.008: (mark-person a15 p5 robo1)  [2.000]
48.009: (move a15 a20 robo1)  [4.000]
52.010: (move-around a20 a19 a18 robo1 pillar2)  [8.000]
60.011: (mark-person a18 p3 robo1)  [2.000]
62.012: (move a18 a13 robo1)  [4.000]
66.013: (move-into a13 a12 robo1 puddle2)  [5.000]
71.014: (move a12 a11 robo1)  [4.000]
75.015: (inject-person a11 p2 robo1)  [6.000]
81.016: (move-into a11 a12 robo1 puddle2)  [5.000]
86.017: (move a12 a17 robo1)  [4.000]
90.018: (move a17 a22 robo1)  [4.000]
94.019: (inject-person a22 p4 robo1)  [6.000]
100.020: (move a22 a17 robo1)  [4.000]
104.021: (move-into a17 a12 robo1 puddle2)  [5.000]
109.022: (move-around a12 a7 a2 robo1 pillar1)  [6.000]
115.023: (push-aside a2 a3 robo1 rubble1)  [6.000]
121.024: (move a2 a3 robo1)  [4.000]
125.025: (move a3 a4 robo1)  [4.000]
129.026: (mark-person a4 p1 robo1)  [2.000]
```

Appendix 3.2: output of Problem file 2 (three robots in different start locations)

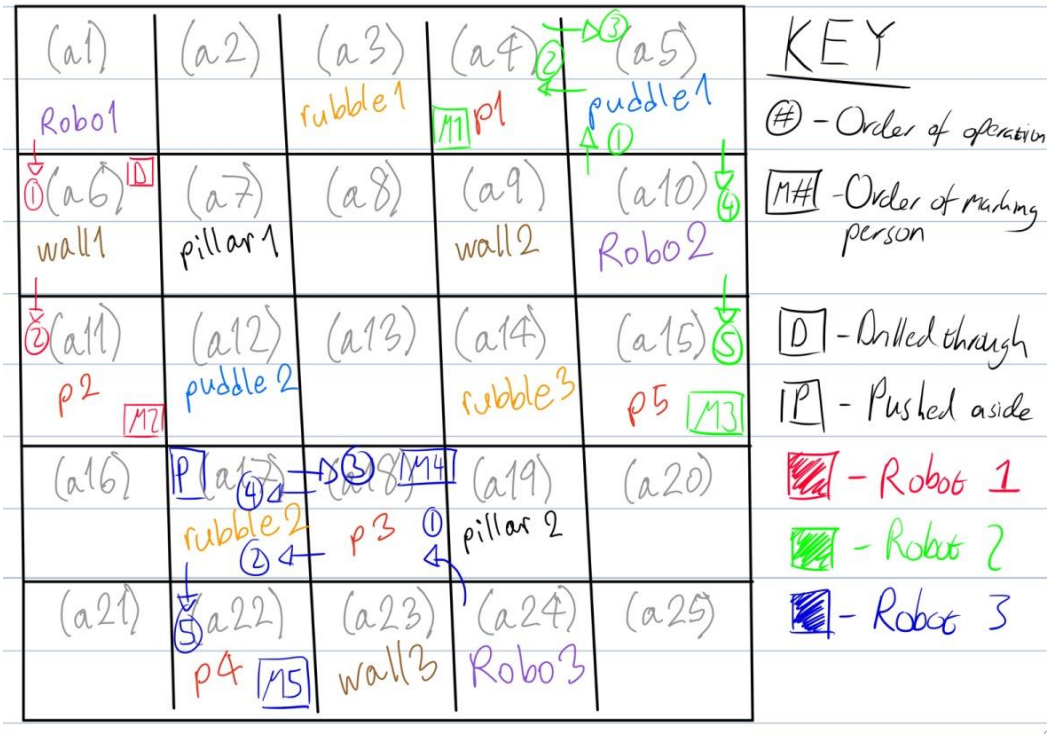
```
Initial heuristic = 28.000, admissible cost estimate 0.000
b (27.000 | 3.000)b (26.000 | 8.000)b (25.000 | 10.000)b (24.000 | 14.000)b (23.000 | 14.000)b (22.000 | 14.000)b (21.000 | 18.002)b (20.00
0 | 18.002)b (19.000 | 18.002)b (18.000 | 18.002)b (17.000 | 24.003)b (16.000 | 24.003)b (15.000 | 24.003)b (14.000 | 24.003)b (13.000 | 24
.003)b (12.000 | 24.003)b (11.000 | 24.003)b (10.000 | 24.003)b (9.000 | 24.003)b (8.000 | 24.003)b (7.000 | 24.003)b (6.000 | 24.004)b (5.
000 | 28.005)b (4.000 | 28.005)b (3.000 | 32.006)b (2.000 | 32.006)b (1.000 | 38.007)(G);;;; Solution Found
; States evaluated: 77
; Cost: 38.007
; Time 0.57
0.000: (move-into a10 a5 robo2 puddle1)  [3.000]
0.000: (move-around a24 a19 a18 robo3 pillar2)  [8.000]
0.000: (drill a1 a6 robo1 wall1)  [10.000]
3.001: (move a5 a4 robo2)  [4.000]
7.002: (mark-person a4 p1 robo2)  [2.000]
8.001: (push-aside a18 a17 robo3 rubble2)  [6.000]
9.003: (move-into a4 a5 robo2 puddle1)  [3.000]
10.001: (move a1 a6 robo1)  [4.000]
12.004: (move a5 a10 robo2)  [4.000]
14.002: (move a18 a17 robo3)  [4.000]
14.002: (move a6 a11 robo1)  [4.000]
16.005: (move a10 a15 robo2)  [4.000]
18.003: (inject-person a11 p2 robo1)  [6.000]
18.003: (move a17 a18 robo3)  [4.000]
20.006: (mark-person a15 p5 robo2)  [2.000]
22.004: (mark-person a18 p3 robo3)  [2.000]
24.005: (move a18 a17 robo3)  [4.000]
28.006: (move a17 a22 robo3)  [4.000]
32.007: (inject-person a22 p4 robo3)  [6.000]
```

Appendix 3.3: output of Problem file 3 (three robots starting from same location)

```
Initial heuristic = 40.000, admissible cost estimate 0.000
b (39.000 | 10.000)b (38.000 | 10.000)b (37.000 | 14.000)b (36.000 | 14.000)b (35.000 | 14.001)b (34.000 | 18.002)b (33.000 | 18.002)b (32.
000 | 18.002)b (31.000 | 18.002)b (30.000 | 18.002)b (29.000 | 18.002)b (28.000 | 18.002)b (27.000 | 22.003)b (26.000 | 22.003)b (25.000 |
22.003)b (24.000 | 22.003)b (23.000 | 22.003)b (22.000 | 22.003)b (21.000 | 22.003)b (20.000 | 22.003)b (19.000 | 22.003)b (18.000 | 22.003
)b (17.000 | 28.004)b (16.000 | 28.004)b (15.000 | 28.004)b (14.000 | 28.004)b (13.000 | 28.005)b (12.000 | 32.006)b (11.000 | 32.006)b (10
.000 | 34.006)b (9.000 | 34.007)b (8.000 | 34.007)b (7.000 | 36.006)b (6.000 | 36.006)b (5.000 | 40.006)b (4.000 | 40.007)b (3.000 | 44.007
)b (2.000 | 44.008)b (1.000 | 50.008)(G);;;; Solution Found
; States evaluated: 98
; Cost: 50.009
; Time 0.57
0.000: (drill a1 a6 robo1 wall1)  [10.000]
0.000: (move a1 a2 robo2)  [4.000]
0.000: (move a1 a2 robo3)  [4.000]
4.001: (move-around a2 a7 a8 robo2 pillar1)  [6.000]
4.001: (push-aside a2 a3 robo3 rubble1)  [6.000]
10.001: (move a1 a6 robo1)  [4.000]
10.002: (move a2 a3 robo3)  [4.000]
10.002: (move a8 a13 robo2)  [4.000]
14.002: (move a6 a11 robo1)  [4.000]
14.003: (move a3 a4 robo3)  [4.000]
14.003: (move a13 a18 robo2)  [4.000]
18.003: (move a11 a16 robo1)  [4.000]
18.004: (move-into a4 a5 robo3 puddle1)  [3.000]
18.004: (mark-person a18 p3 robo2)  [2.000]
20.005: (move-around a18 a19 a20 robo2 pillar2)  [8.000]
21.005: (move a5 a4 robo3)  [4.000]
22.004: (move a16 a11 robo1)  [4.000]
25.006: (mark-person a4 p1 robo3)  [2.000]
26.005: (inject-person a11 p2 robo1)  [6.000]
28.006: (move a20 a15 robo2)  [4.000]
32.006: (move a11 a16 robo1)  [4.000]
32.007: (mark-person a15 p5 robo2)  [2.000]
36.007: (move a16 a21 robo1)  [4.000]
40.008: (move a21 a22 robo1)  [4.000]
44.009: (inject-person a22 p4 robo1)  [6.000]
```

Appendix 4.1: Path of Problem file 1 (one robot)

Appendix 4.2: Path of Problem file 2 (three robots in different start locations)



Appendix 4.3: path of Problem file 3 (three robots starting from same location)