LONDON'S GLOBAL UNIVERSITY

COMP0037

Report

Planning in Uncertain Worlds

Group AS

Student Name	Student number
Arundathi Shaji Shanthini	16018351
Dmitry Leyko	16021440
Tharmetharan Balendran	17011729

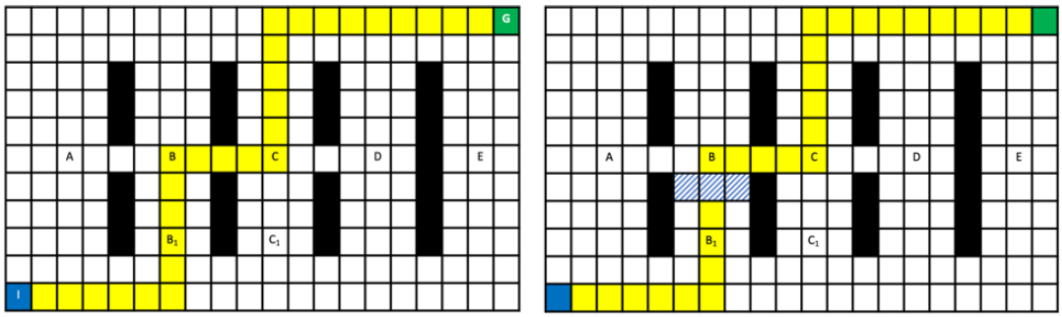
Department: Department of Electronic and Electrical Engineering

Submission Date: 28th of April 2020

Contents

1 Policy Selection in a Dynamic Environment

1.1 Policy Selection when Obstacle is Observed



(a) The original planned path form I to G going through Aisle B and C.

(b) An obstacle in aisle B obstructs the planned path of the robot.

Figure 1: Illustration of case where robot observes an obstruction to it's planned path.

The scenario that we will be analysing is the case shown in Fig. 1a. The robot is required to to plan a path from cell I to cell G that are marked as blue and green in Fig. 1a respectively. This figure also shows the path originally planned by the robot and it can be seen that it goes down via aisle B assuming it has no knowledge of any dynamic obstacles that may be present in the environment yet. However, once the robot turns into aisle B and reaches cell B_1, it observes that the aisle is blocked. At this point, the robot can either decide to wait until the obstruction clears or it can re-plan its path.
If the robot is to wait, the time that the robot must wait for the obstacle to clear [1] is represented by Eq. (1) as:

$$T = \frac{0.5}{\lambda_B} + \widetilde{T} \tag{1}$$

The wait time is dependent on λ_B and a random variable $\widetilde{T}$. The random variable $\widetilde{T}$ is sampled from an exponential distribution with a rate parameter of $2\lambda_B$. The probability density function (PDF) for $\widetilde{T}$ is shown in Eq. (2).

$$f(t) = \begin{cases} \lambda e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{2}$$

where, the rate parameter $\lambda = 2\lambda_B$.

As previously mentioned, the robot has two options to choose from: to wait for the obstacle to clear, or to re-plan and execute the new path. These are the two different policies the robot must choose from. We use the symbol π to denote a policy. A policy is a mapping from the world state to an action set the robot can execute, given by:

$$\pi : \mathbf{X} \rightarrow \mathbf{U} \tag{3}$$

where, π is the symbol used to denote a policy. Eq. (3)

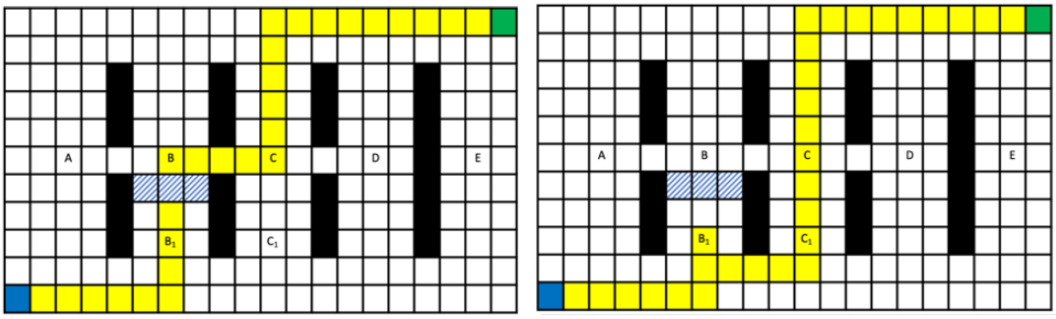
We know the robot detects the obstacle when it reaches cell $B1$. Let us assume that to reach cell $B1$ it took K_O stages. If the robot chooses to wait for the obstacle to clear, then lets say this adds K_1 stages. On the other hand, if it chooses to re-plan the route via aisle C then lets say that this would add

[1] This time is calculated from the moment the obstacle was detected

K_2 stages. Following these notations, the policy for the robot to wait is denoted as π_K^1 and the policy for re-planning is denoted as π_K^2 in this section. Note that, these policies are padded out with zero-cost actions up till K_1/K_2 stages and only account for actions $\mathbf{u}_K^1$ and $\mathbf{u}_K^2$ that are to be executed further.

To decide which policy is better on average, the robot considers the expected value of the cost function for all the different policies and chooses the one with the lowest cost. In the given case, for policy π_K^1 to be chosen, the inequality characterised by Eq. (4) must be true.

$$\mathbb{E}\left[L\left(\pi_K^1\right)\right] \leq \mathbb{E}\left[L\left(\pi_K^2\right)\right] \tag{4}$$



(a) The path for the wait policy π_K^1. The robot will wait at the cell marked B_1. The dashed blue cells represent the obstacle.

(b) The path for the re-plan policy π_K^2 which bypasses aisle B and goes down aisle C.

Figure 2: Illustration of two different policies the robot has to choose from.

We can see from Fig. 1a that the cost of the original planned path is given by the expression in Eq. (5). In this equation, the terms L_{XY} denote the cost of the shortest path between cell X and cell Y. Additionally, the term L_W represents the non-zero cost of the action $\boldsymbol{u}_w$. This action is state preserving (i.e. $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_w) = \boldsymbol{x}_k$) and is used to represent the action of the robot waiting for one unit of time.
Therefore, the cost of waiting for the obstacle to clear is:

$$L\left(\pi_K^1\right) = L_{IB_1} + TL_W + L_{B_1B} + L_{BC} + L_{CG} \tag{5}$$

Similarly from Fig. 2b which shows the re-planned path, we can derive the cost of this path as:

$$L\left(\pi_K^2\right) = L_{IB_1} + L_{B_1C_1} + L_{C_1C} + L_{CG} \tag{6}$$

Now, by substituting Eq. (5) and Eq. (6) into Eq. (4), we obtain the inequality shown in Eq. (7).

$$\begin{aligned} \mathbb{E}\left[L_{IB_1} + TL_W + L_{B_1B} + L_{BC} + L_{CG}\right] &\leq \mathbb{E}\left[L_{IB_1} + L_{B_1C_1} + L_{C_1C} + L_{CG}\right] \\ L_{IB_1} + \mathbb{E}\left[T\right]L_W + L_{B_1B} + L_{BC} + L_{CG} &\leq L_{IB_1} + L_{B_1C_1} + L_{C_1C} + L_{CG} \\ \mathbb{E}\left[T\right] &\leq \frac{L_{B_1C_1} + L_{C_1C} - L_{B_1B} - L_{BC}}{L_W} \end{aligned} \tag{7}$$

The quantity $\mathbb{E}\left[T\right]$ is the expected mean value of the time the robot has to wait for the obstacle to clear. We can compute the expected value for this wait time as the mean of the function represented by Eq. (1). Then, the expected value for the time taken can be computed as follows:

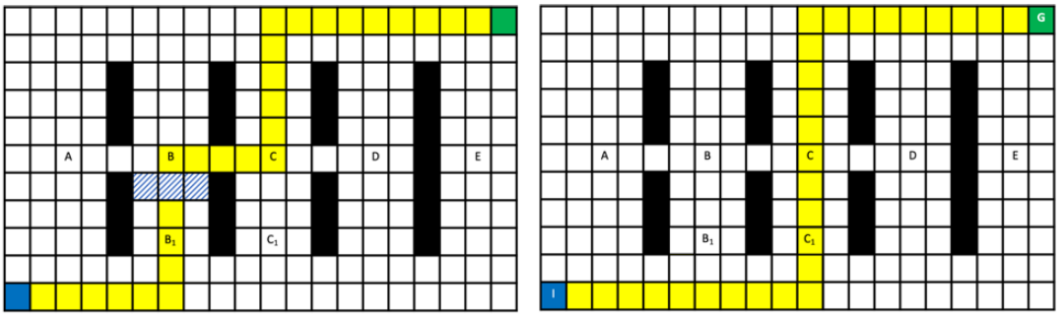
$$
\begin{aligned}
\mathbb{E}[T] &= \mathbb{E}\left[\frac{0.5}{\lambda_B} + \widetilde{T}\right] \\
&= \frac{0.5}{\lambda_B} + \mathbb{E}\left[\widetilde{T}\right] \\
&= \frac{0.5}{\lambda_B} + \int_0^\infty 2\lambda_B t e^{-2\lambda_B t} dt \\
&= \frac{0.5}{\lambda_B} + 2\lambda_B \left[(t)\left(-\frac{1}{2\lambda_B} e^{-2\lambda_B t}\right) + \frac{1}{2\lambda_B}\int_0^\infty e^{-2\lambda_B t} dt\right]_0^\infty \\
&= \frac{0.5}{\lambda_B} + 2\lambda_B \left[(t)\left(-\frac{1}{2\lambda_B} e^{-2\lambda_B t}\right) - \left(\frac{1}{2\lambda_B}\right)^2 e^{-2\lambda_B t}\right]_0^\infty \\
&= \frac{0.5}{\lambda_B} - \left[t e^{-2\lambda_B t} + \frac{1}{2\lambda_B} e^{-2\lambda_B t}\right]_0^\infty \\
&= \frac{0.5}{\lambda_B} - (0 + 0 - 0 - \frac{1}{2\lambda_B}) = \frac{0.5}{\lambda_B} + \frac{1}{2\lambda_B} = \frac{1}{\lambda_B}
\end{aligned}
\tag{8}
$$

Substituting the expression from Eq. (8) into Eq. (7) we obtain the inequality in Eq. (9). This inequality also takes into consideration the constraint that $\lambda_B > 0$ and negates the solution when $\lambda_B < 0$.

$$
\begin{aligned}
\frac{1}{\lambda_B} &\leq \frac{L_{B_1C_1} + L_{C_1C} - L_{B_1B} - L_{BC}}{L_W} \\
\lambda_B &\geq \frac{L_W}{L_{B_1C_1} + L_{C_1C} - L_{B_1B} - L_{BC}}
\end{aligned}
\tag{9}
$$

Therefore, from Eq. (9) we get that the smallest possible value for λ_B for which the waiting policy π_K^1 is a better option than the re-plan policy π_K^2 is $\frac{L_W}{L_{B_1C_1} + L_{C_1C} - L_{B_1B} - L_{BC}}$.

1.2 Policy Selection at Start



(a) The scenario where the robot decides to go down Aisle B, encounters an obstacles and waits for it to clear.

(b) The scenario where the robot decides to avoid Aisle B completely due to the obstacle.

Figure 3: The different policies the robot can pick from at the beginning.

In the previous scenario, the robot reacted to the obstacle after detecting it upon reaching cell B_1. However, if the robot has prior knowledge of the probability that an obstacle might be present at aisle B, then the robot may make the decision of whether it should avoid aisle B right at the start before it moves. So given that, the robot has knowledge of where there are potential obstacles and the associated probability distribution for the wait times at these obstacles, a decision can be made at the start on whether or not

to avoid that aisle. This could mean that depending on the probabilities involved, the robot may decide to avoid the obstacle altogether instead of going through the route and having to react. In the case of the warehouse example with 5 aisles, and one obstacle in aisle B, the two policies that the robot can choose from are: travel down aisle B and wait if an obstacle is present (illustrated in Fig. 3a) or to avoid aisle B and plan directly via aisle C (illustrated in Fig. 3b). This approach is favourable especially as it avoids the robot from having to traverse the extra distance to the cell marked B_1 in the case when waiting at obstacle B is a more unfavourable option.

Similar to the approach in §1.1, we may denote the policy of going down aisle B and waiting as π_K^1 and the policy of going down aisle C as π_K^2. The two two possible paths that represent these two policies are shown in Fig. 3. The costs of these different policies can be defined as:

- Policy π_K^1 : Drive down aisle B and wait for the obstacle to clear. The cost of this scenario is given by the expression in Eq. (10).

$$L(\pi_k^1) = L_{IB_1} + TL_W + L_{B_1B} + L_{BC} + L_{CG} \tag{10}$$

- Policy π_K^2 : Completely avoid aisle B and traverse directly to the goal via aisle C. It is known that no obstacle exists on this aisle. The cost for this policy is given in Eq. (11).

$$L(\pi_k^2) = L_{IC} + L_{CG} \tag{11}$$

As we are concerned with the average case scenario we compare the expected value of the loss function just as we did in §1.1. Policy π_K^2 will be chosen over policy π_K^1 if the expected value of it's cost is lower as shown by Eq. (12).

$$\mathbb{E}\left[L\left(\pi_K^2\right)\right] \leq \mathbb{E}\left[L\left(\pi_K^1\right)\right] \tag{12}$$

Then, substituting the loss functions from Eq. (10) and (11) into the inequality in Eq. (12), we obtain the result shown in Eq. (13).

$$\begin{aligned} \mathbb{E}[L_{IC} + L_{CG}] &\leq \mathbb{E}[L_{IB_1} + TL_W + L_{B_1B} + L_{BC} + L_{CG}] \\ \mathbb{E}[T]L_W &\geq L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC} \\ \mathbb{E}[T] &\geq \frac{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}}{L_W} \end{aligned} \tag{13}$$

By substituting the expected value for T found in Eq. (8) we obtain a constraint for λ_B as shown in Eq. (14). Once again the solutions for $\lambda_B < 0$ have been ignored.

$$\begin{aligned} \frac{1}{\lambda_B} &\geq \frac{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}}{L_W} \\ \lambda_B &\leq \frac{L_W}{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}} \end{aligned} \tag{14}$$

Therefore from the above inequality we can say that the largest value for λ_B for which the robot will decide to go directly down aisle C avoiding aisle B is $\dfrac{L_W}{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}}$.

1.3 Considering the Probability of the Obstacle Being Present

In reality, the obstacle would not be present at aisle B all the time. This can be taken into account using a probability, say p_B, associated with the scenario that the obstacle is present[2]. On incorporating these probabilities, the mean expected wait time weighted by the probability that the obstacle is present would therefore change from what was derived as Eq. (8) to $p_B \cdot \mathbb{E}[T]$. On the other hand, in case the obstacle isn't present, the robot does not have wait and therefore the mean wait time is 0. By taking the sum of these expected wait times weighted by their probabilities of occurring, we obtain the expected wait time that takes into consideration the probability of the obstacle being present as shown below in Eq. (15):

$$\begin{aligned} \mathbb{E}_B[T] &= p_B \cdot \mathbb{E}[T] + (1 - p_B) \cdot (0) \\ &= p_B \cdot \frac{1}{\lambda_B} = \frac{p_B}{\lambda_B} \end{aligned} \tag{15}$$

where, $\mathbb{E}[T]$ is time that the robot is expected to wait given the obstacle is present at aisle B and it is given by Eq. (7)

Eq. (12) and Eq. (15) represents the inequality based on which the robot would choose the policy to plan straight through aisle C by avoiding aisle B. So for the robot to chose aisle B, the inequality given by Eq. (13) must be reversed and by substituting the new modified value for wait time that incorporates the probability of the obstacle being there given by Eq. (15), we get:

$$\begin{aligned} \mathbb{E}_B[T] &\leq \frac{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}}{L_W} \\ \frac{p_B}{\lambda_B} &\leq \frac{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}}{L_W} \end{aligned} \tag{16}$$

Assuming that the rate parameter λ_B is constant, we arrive at the equation shown in Eq. (17).

$$p_B \leq \frac{\lambda_B \left(L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}\right)}{L_W} \tag{17}$$

Therefore, for a fixed value of λ_B the value of p_B below which the robot will attempt to drive aisle B first is $\frac{\lambda_B \left(L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}\right)}{L_W}$.

1.4 Considering Multiple Obstacles

Now, let us consider a scenario where both aisles B and C have obstacles on them. The wait time associated with the obstacle in aisle B to clear, is sampled from the distribution outlined in Eq. (1). The probability that an obstacle is present in aisle B is given as p_B. Similarly, the obstacle in aisle C has a wait time distribution identical to that of the obstacle in aisle B, but with a characterizing parameter of λ_C. Further, the probability that obstacle C is present at aisle C is given by the probability p_C. Given this information, the expected wait times for obstacle B and obstacle C are as follows:

$$\mathbb{E}_B[T] = \frac{p_B}{\lambda_B} \tag{18}$$

$$\mathbb{E}_C[T] = \frac{p_C}{\lambda_C} \tag{19}$$

Suppose the robot has the same start and goal cells as described in §1.1 and has to choose the policy at the beginning, there are 6 different key scenarios (all of which are illustrated in Fig. 4) that are worth considering:

[2]This means that the probability that the obstacle is absent can be given by $(1 - p_B)$

- Policy π_K^1 : The robot goes down aisle B and observes an obstacle. It then decides to wait for the obstacle to clear. The expected cost for this policy is given by:

$$\mathbb{E}\left[L\left(\pi_K^1\right)\right] = L_{IB_1} + \mathbb{E}_B\left[T\right] L_W + L_{B_1B} + L_{BC} + L_{CG} \tag{20}$$

- Policy π_K^2 : The robot goes down aisle B and observes an obstacle so re-plans down aisle C. While travelling down aisle C, it observes another obstacle and waits for it to clear before continuing to the goal. The expected cost for this policy is given by:

$$\mathbb{E}\left[L\left(\pi_K^2\right)\right] = L_{IB_1} + L_{B_1C_1} + \mathbb{E}_C\left[T\right] L_W + L_{C_1C} + L_{CG} \tag{21}$$

- Policy π_K^3 : The robot goes down aisle B and observes an obstacle so re-plans down aisle C. While travelling down aisle C, it observes another obstacle, so it re-plans down aisle D which is known to have no obstacles. The expected cost for this policy is given by:

$$\mathbb{E}\left[L\left(\pi_K^3\right)\right] = L\left(\pi_K^3\right) = L_{IB_1} + L_{B_1C_1} + L_{C_1} + L_{DG} \tag{22}$$

- Policy π_K^4 : The robot goes down aisle C and observes an obstacle, so it waits for the obstacle to clear before continuing to the goal. The expected cost for this policy is given by:

$$\mathbb{E}\left[L\left(\pi_K^4\right)\right] = L_{IC_1} + \mathbb{E}_C\left[T\right] L_W + L_{C_1C} + L_{CG} \tag{23}$$

- Policy π_K^5 : The robot goes down aisle C and observes an obstacle so it re-plans down aisle D which is known to have no obstacles. The expected cost for this policy is given by:

$$\mathbb{E}\left[L\left(\pi_K^5\right)\right] = L\left(\pi_K^5\right) = L_{IC_1} + L_{C_1D} + L_{DG} \tag{24}$$

- Policy π_K^6 : The robot directly goes down aisle D which is known to have no obstacles and avoids both aisles B and C. The expected cost for this policy is given by:

$$\mathbb{E}\left[L\left(\pi_K^6\right)\right] = L\left(\pi_K^6\right) = L_{ID} + L_{DG} \tag{25}$$

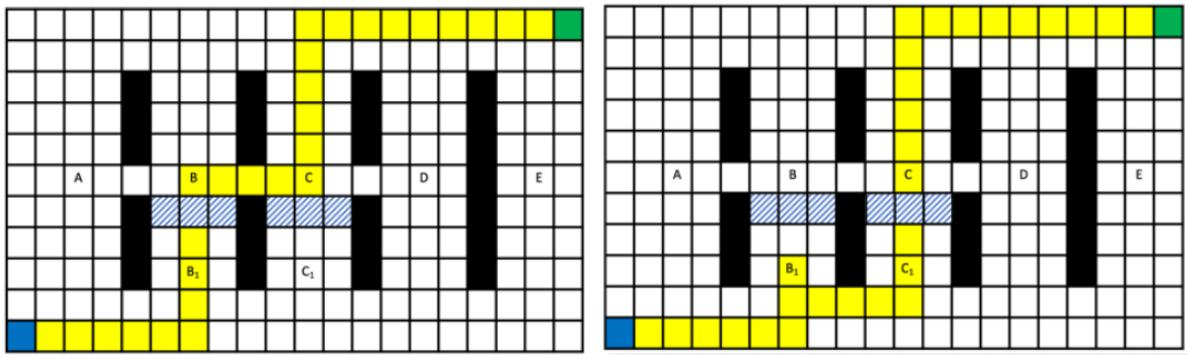
Note that, here we ignore the cases of going down aisle A and aisle E as these will inherently have a larger path length due to the topology of the example scenario. Additionally, back-tracking of aisles is also ignored (e.g. going from aisle C back to aisle B) as this would also have larger costs than those considered above.

We are given that the robot chooses policy π_K^6 by going straight down aisle D. This suggests that the expected cost of this policy is lower than any other policy available to the robot as suggested by Eq. (26)

$$\begin{aligned} \mathbb{E}\left[L\left(\pi_K^6\right)\right] &= \min_{n=1,\dots,6} \mathbb{E}\left[L\left(\pi_K^n\right)\right] \\ \mathbb{E}\left[L\left(\pi_K^6\right)\right] &\leq \min_{n=1,\dots,5} \mathbb{E}\left[L\left(\pi_K^n\right)\right] \end{aligned} \tag{26}$$

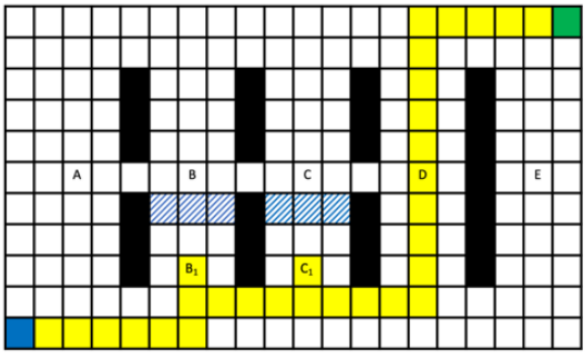
We can see that some of these policies have common terms. For example, we may have a look at policies π_K^2 and π_K^4. π_K^4 will be picked over π_K^2 if the following inequality holds:

$$\begin{aligned} \mathbb{E}\left[L\left(\pi_K^4\right)\right] &\leq \mathbb{E}\left[L\left(\pi_K^2\right)\right] \\ L_{IC_1} + \mathbb{E}_C\left[T\right] L_W + L_{C_1C} + L_{CG} &\leq L_{IB_1} + L_{B_1C_1} + \mathbb{E}_C\left[T\right] L_W + L_{C_1C} + L_{CG} \\ L_{IC_1} &\leq L_{IB_1} + L_{B_1C_1} \end{aligned} \tag{27}$$

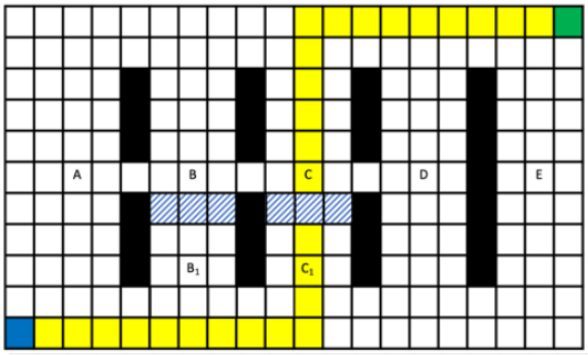


(a) Illustration of Policy π_K^1

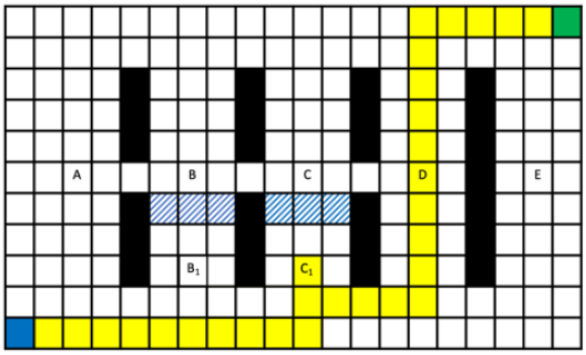
(b) Illustration of Policy π_K^2



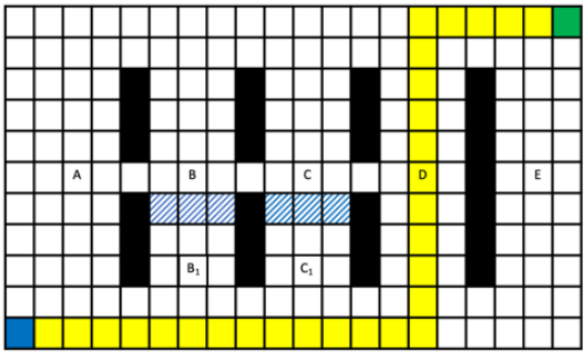
(c) Illustration of Policy π_K^3



(d) Illustration of Policy π_K^4



(e) Illustration of Policy π_K^5



(f) Illustration of Policy π_K^6

Figure 4: The diagrams illustrate the 6 different policies that are considered. The dashed blue areas indicate temporary obstacles. The yellow paths represent the planned path produced by the policy under question. All planned paths are from the same start cell (blue) to the same goal cell (green).

As we know that L_{IC_1} represents the minimum path cost from I to C_1, the inequality in Eq. (27) is true. This results in the robot always picking π_K^4 over π_K^2. We can therefore ignore policy π_K^2. The exact argument involving an inequality similar to Eq. (27), can be used to show that the robot will always pick π_K^6 over π_K^3 and π_K^5. Therefore, we can also ignore policies π_K^3 and π_K^5. We therefore now have, simplified the inequality given by Eq. (26) as shown in Eq. (28).

$$\begin{aligned} \mathbb{E}\left[L\left(\pi_K^6\right)\right] &= \min_{n=1,4,6} \mathbb{E}\left[L\left(\pi_K^n\right)\right] \\ \mathbb{E}\left[L\left(\pi_K^6\right)\right] &\leq \min_{n=1,4} \mathbb{E}\left[L\left(\pi_K^n\right)\right] \end{aligned} \tag{28}$$

We now attempt to try and reduce this further by similar comparisons. However, lack of knowledge of the values of λ_B and λ_C or the relationship between them, prevents us from reducing or drawing comparison between these costs further. However, we would like highlight in advance that the attempt below involves combining 2 inequalities to give an absolute upper bound value which may not represent a policy (or path) that has a meaning in reality.
We start by comparing the policies π_K^1 and π_K^6. If π_K^6 is chosen between the two, then the following inequality must hold:

$$L\left(\pi_K^6\right) \leq L_{IB_1} + \mathbb{E}_B\left[T\right] L_W + L_{B_1B} + L_{BC} + L_{CG} \tag{29}$$

Similarly, to choose π_K^6 over π_K^4 the inequality in Eq. (30) must hold:

$$L\left(\pi_K^6\right) \leq L_{IC_1} + \mathbb{E}_C\left[T\right] L_W + L_{C_1C} + L_{CG} \tag{30}$$

The maximum value of $L\left(\pi_K^6\right)$ for policy π_K^6 to be chosen is either given by Eq. (29) or by Eq. (29) whichever one is smaller. The inequalities cannot be compared to find the exact maximum value for the maximum path cost. However, by using logic, we may derive an upper bound for the path cost of π_K^6. From Eq. (29) and Eq. (30) we can obtain the inequality shown in Eq. (31) which will also hold. It should be noted that this inequality does not express the maximum value for $L\left(\pi_K^6\right)$ for which policy π_K^6 is chosen but rather the upper bound estimation for that value. The actual upper bound of the cost of policy π_K^6 is expressed in Eq. (28) as π_K^1 or π_K^4 whichever is smaller.

$$\begin{aligned} L\left(\pi_K^6\right) &\leq L_{IB_1} + \mathbb{E}_B\left[T\right] L_W + L_{B_1B} + L_{BC} + L_{CG} + L_{IC_1} + \mathbb{E}_C\left[T\right] L_W + L_{C_1C} \\ L\left(\pi_K^6\right) &\leq L_{IB_1} + L_W(\frac{p_B}{\lambda_B} + \frac{p_C}{\lambda_C}) + L_{B_1B} + L_{BC} + L_{CG} + L_{IC_1} + L_{C_1C} \end{aligned} \tag{31}$$

Values for $\mathbb{E}_B\left[T\right]$ and $\mathbb{E}_C\left[T\right]$ were taken from Eq. (19). Here we use the fact that if $A < B + C$ is true and $A < B + D$ is true then it must also follow that $A < B + C + D$.

2 Implementation of the System in ROS

This section discusses the implementation of the principles discussed in §1. The simulations were carried out using the STDR Simulator running on Robotics Operating System (ROS). The nodes and all supporting scripts were coded in Python.

2.1 Planning Path via an Aisle

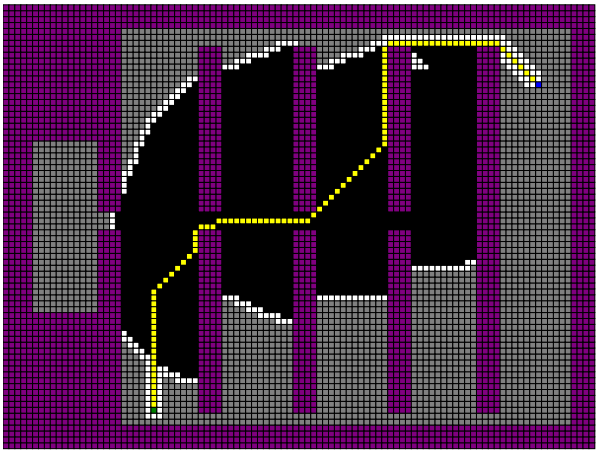
Figure 5: The map representing the environment where the robot will traverse. Solid black objects represent permanent obstacles while grey obstacles represent temporary obstacles. The darker the obstacle appears, the higher the associated value of λ. The wait time from time of being observed is given by Eq. (1). The referencing of aisles will be the same as in §1 with aisles A to E from left to right.

The map in Fig. 5 shows the map of the environment that was used. The map consists of 5 aisles. The start cell and goal cell throughout this section will be the same and al discussions will be for this single case. The start cell is in the bottom of aisle A while the goal cell is in the top of aisle E. The map in Fig. (5) implies that there are 4 temporary obstacles on the map. However only one of them will be active throughout the simulations carried out in this section. This is the obstacle in aisle B.

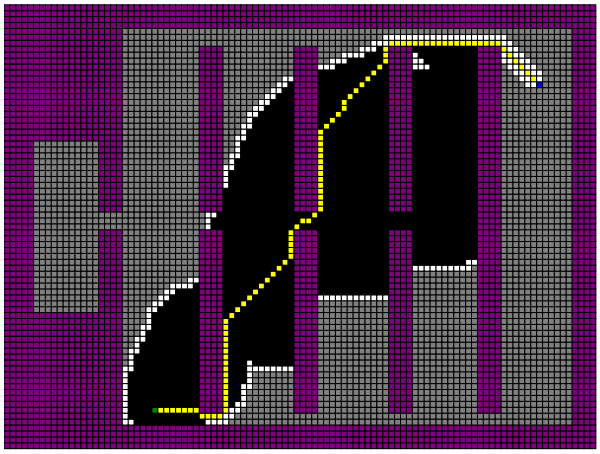
As we are concerned with comparing the costs of paths down specific aisles, a function needs to be implemented to allow the planning of a path down a chosen aisle. This is accomplished by the `planPathToGoalViaAisle` method in the `ReactivePlannerController` class. The method takes 4 parameters of which 1 is optional:

- `startCellCoords` : The coordinates of the start cell in the search grid (different from the actual coordinates of the cell on the map)
- `goalCellCoords` : The coordinates of the goal cell in the search grid (different from the actual coordinates of the cell on the map)
- `aisle` : The aisle down which a path is desired
- `graphics = True` : Optional parameter that takes a default value of True. Ensures that intermediate searches don't result in constant updating of the search grid graphics. Instead the search grid graphics are only updated during the computation of the path to be traversed by the robot.

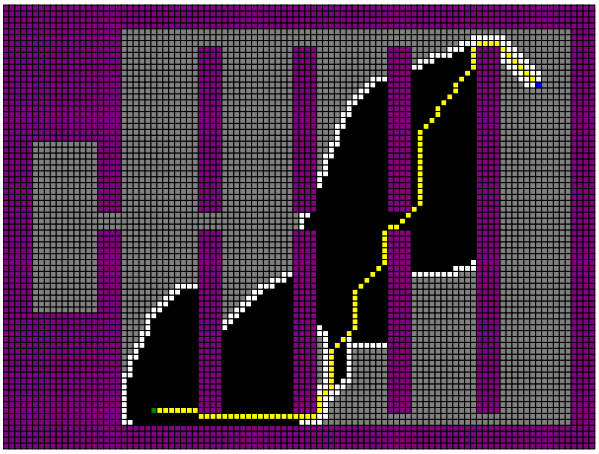
The chosen approach to plan a route down a specific aisle is to split the path planning problem into two parts. In the first part the chosen planner (in this case an A* algorithm with an octile heuristic was implemented by the `AStarPlanner` class) will plan a path from the start cell (defined by `startCellCoords`)to a cell in the chosen aisle. In the next part the planner will plan a part from the cell in the chosen aisle to the goal cell defined by `goalCellCoords`. Having obtained these two paths, the next step is to concatenate the two paths to produce the planned path via the chosen aisle. The cells of each aisle are hard coded and returned by a method `getAisleMidpoint` in the `ReactivePlannerController` class. The method takes a single parameter which is the chosen aisle.



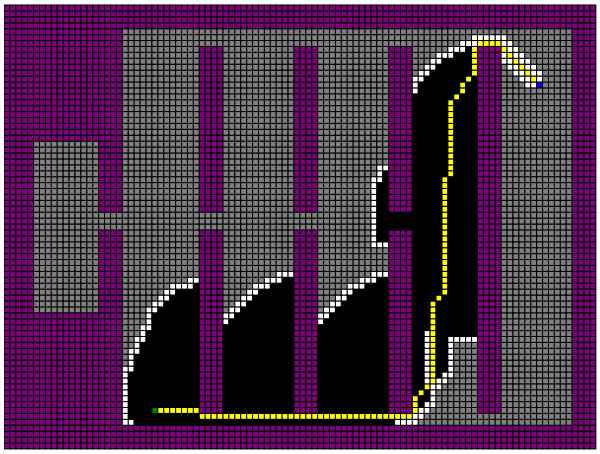
(a) Search grid for a path planned via Aisle A



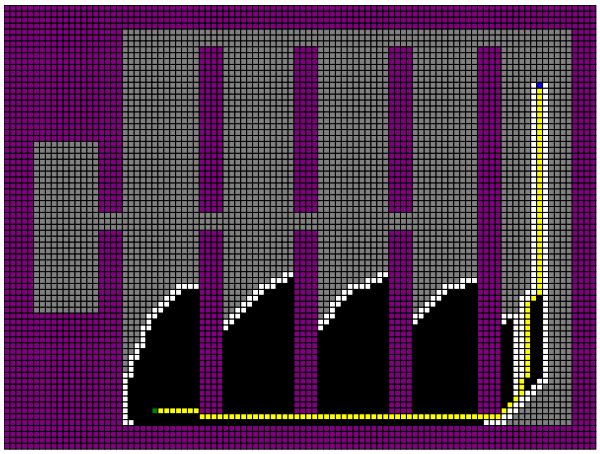
(b) Search grid for a path planned via Aisle B



(c) Search grid for a path planned via Aisle C



(d) Search grid for a path planned via Aisle D



(e) Search grid for a path planned via Aisle E

Figure 6: Search grids for the paths planned from the start (green) to the goal (blue) via a specific aisle. Purple cells indicate obstacles. White cells and Black cells represent alive and dead cells for the search algorithm respectively. Grey cells are unoccupied cells that are unvisited by the search algorithm.

To concatenate the two paths the `addToEnd` method of the `PlannedPath` class is used. This method takes a planned path as an argument and adds the waypoints of this planned path to the end of the planned path on which the method is called. It also sums the travel cost and path cardinality.

For the plotting of search grids the two separate search grids from the two executions of the search algorithm need to be merged. This was achieved by the addition of the `leftMergeGrid` method in the `SearchGrid` class. This method acts upon a search grid and takes the cell grid of another search grid as a parameter. The search grid taken as the parameter is assumed to be the first search grid corresponding to the path planned from the start to the aisle cell. The start and goal cells are adjusted accordingly and dead/alive are transferred to the latter search grid as well. Finally, the complete planned path was plotted onto the search grid in yellow and then displayed to the user if graphics is enabled. The resulting search grid of paths planned down each aisle is shown Fig. 6.

2.2 Waiting for the Obstacle to Clear

As discussed in §1, the robot may choose to wait for an obstacle to clear. During this time the robot should halt it's execution of the current path and wait for the blocking obstacle to clear. This is implemented again in the `ReactivePlannerController` class under the `waitUntilTheObstacleClears` method. This method takes two parameters which are `startCellCoords` and `goalCellCoords` (the cell coordinates for the start cell and goal cell respectively). The waiting functionality is achieved by utilizing `rospy.sleep` fucntion provided by ROS. The method checks if the current path is still traversable using the `checkIfPathCurrentPathIsStillGood` method of the `ReactivePlannerController` class. While this method returns `False`, the `waitUntilTheObstacleClears` method sleeps for 0.5s in simulation time before rechecking. The elapsed wait time is also computed and if the wait time exceeds the threshold (a set maximum) then, the robot will stop waiting and re-plans via another aisle chosen by the `chooseAisle` method. This threshold value is defined by the `maxWaitTime` attribute of the `ReactivePlannerController` class which in turn is read from the ROS parameter `max_wait_time`.

Once a robot encounters an obstacl,e it must decide if it is more cost efficient to re-plan via another aisle or wait for the obstacle to clear so that it can continue on its planned route. This is implemented in the `ReactivePlannerController` class by the `shouldWaitUntilTheObstacleClears` method. This method returns `True` if it is more cost effective to wait and `False` if the robot should re-plan the path. To make a decision on which policy is better, the expected cost of both policies need to be computed. To compute the cost of replanning we simply call the previously implemented `planPathToGoalViaAisle` with the desired aisle determined by the `chooseAisle` method. This returns a `PlannedPath` object of which the `travelCost` attribute stores the path cost. However, computation of the waiting policy cost is slightly more troublesome. As we do not have access to the obstacle-free map within the `ReactivePlannerController` class, we need a workaround to compute the path cost of the current cell to the goal cell via the blocked aisle. We do however, have access to the planned path that the robot embarked on in the `currentPlannedPath` attribute. This includes the section of the path already traversed by the robot. To accommodate for this, a temporary `AStarPlanner` object was used to plan the path from the current cell to the start cell. By subtracting the cost of this path from the cost of the path stored in `currentPlannedPath` we obtain the desired path cost.

Finally, the expected wait cost needs to be added. This is computed by multiplying the cost of waiting one unit of time (L_W) by the expected wait time. The expected wait time is computed by the expression found in Eq. (8) and is characterized by the value of λ_B. To make a decision as to which policy to choose, the algorithm simply picks the one with the lowest cost: if the cost of waiting is less than the cost of replanning, `shouldWaitUntilTheObstacleClears` returns `True` otherwise it returns `False`. Both the value of λ_B and the value of L_W are read from ROS parameters and can be customized in the launch scripts.

For one instance when the simulation was run, the initially chosen aisle was set to aisle B. When the robot encountered the obstacle, it computed that the path cost of the replanning path was 103.74 while the path cost of the waiting plan was 98.67. By considering the expression found in Eq. (9) we can compute the threshold λ_B value under which the robot will always move if it encounters the obstacle. This value is found as shown in Eq. (32). The cost of waiting one unit of time (L_W) is taken to be 2.

$$\begin{aligned} \lambda_B &\leq \frac{L_W}{PathCost_{Replan} - PathCost_{Wait}} \\ \lambda_B &\leq \frac{2}{103.74 - 98.67} \\ \lambda_B &\leq 0.3945 \end{aligned} \tag{32}$$

2.3 Policy Selection at Start

Sometimes it is preferred that the robot decides which aisle is most favourable right at the start before it starts to move on any planned path. This approach is discussed in §1.2 and this section discusses the implementation of the same. The robot will utilize prior knowledge of the environment and the nature/location of the obstacles to make an informed decision as to which aisle minimizes the average wait time. The nature of the environment is fully characterized by the map available to the robot (shown in Fig. 5). The location of the obstacles is fixed: there is one obstacle in aisle B. The nature of this obstacle is fully characterized by the probability of the obstacle being present p_B and the parameter λ_B. From these two parameters the expected wait time of the obstacle can be found to be equal to $\frac{p_B}{\lambda_B}$ as shown in Eq. (15).

Policy selection at the beginning was achieved by modifying the `chooseInitialAisle` method of the `ReactivePlannerController` class. If a particular aisle is favourable by the operator then this can be set as a ROS parameter in the launch script and the robot will choose this aisle as the first aisle. However, if the ROS parameter does not exist, the algorithm will proceed to plan a path to the goal via each of the available aisles and compare the policy costs. The planning of paths via a specified aisle is done by calling the previously implemented `planPathToGoalViaAisle` method. Once the paths are planned, the path costs of each of the paths are stored. Then, the expected cost of waiting is added to the path cost of aisle B by implementing the equation shown in Eq. (33). The cost of waiting one unit of time L_W is taken to be 2 and the parameter p_B is taken to be 0.8. All of these parameters, as well as λ_B can be set in the launch files as they are read from ROS parameters.

$$\mathbb{E}[L_{wait}] = L_W \cdot \frac{p_B}{\lambda_B} \tag{33}$$

Once the policy costs of the paths via each aisle has been computed, the best policy is chosen as the one with the lowest cost. For one instance of running the STDR simulation in ROS, we obtained the path costs for paths via each of the aisles shown in Table 1. We can see that the shortest two paths are the one via aisle B and aisle C. Even though the path via aisle B is the shorter one, in the case when $\lambda_B = 1$, the policy cost of the path via aisle B becomes larger than that via aisle C. We can compute the threshold value of λ_B using the expression found in Eq. (16) and from this we can derive Eq. (34) which is the inequality that shows the constraint of λ_B for which the robot will plan directly via aisle C. As mentioned earlier, p_B is taken to 0.8 and L_W is taken to be 2 here.

$$\begin{aligned}\lambda_B &\leq \frac{p_B \cdot L_W}{PathCost_{aisleC} - PathCost_{aisleB}} \\ \lambda_B &\leq \frac{2 \cdot 0.8}{115.50 - 114.91} \\ \lambda_B &\leq 2.72\end{aligned} \tag{34}$$

Aisle	Path Cost	Policy Cost
A	116.43	116.43
B	114.91	116.51
C	115.50	115.50
D	124.28	124.28
E	117.31	117.31

Table 1: Path cost of path planned to goal from the start via each of the available aisles. The policy costs are also given. $\lambda_B = 1$ and $p_B = 0.8$ in this scenario.