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

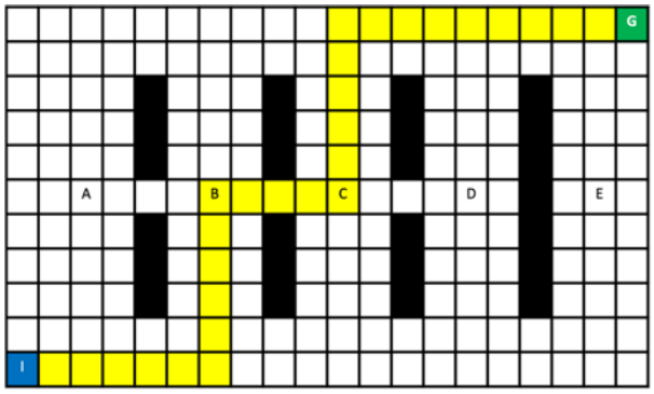


Figure 1: The original planned path form I to G going through Aisle B and C.

The scenario that we will be analysing is the case shown in Fig. 1. The robot is required to go from a cell I to a cell G. These cells are marked blue and green in Fig. 1 respectively. The figure also shows the original planned path that the robot computed and this goes down aisle B. However, once the robot turns into aisle B it detects that the aisle is blocked. This observation is done at the point when the robot reaches the cell labelled B_1. At this point the robot can decide either to wait until the obstruction clears or it can re-plan a path.
If the robot decides to wait, the time the robot must wait for the obstacle to clear, after the robot detects it may be represented by Eq. (1).

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

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 given by 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. A policy is a mapping from the world state to an action the robot can execute, given by:

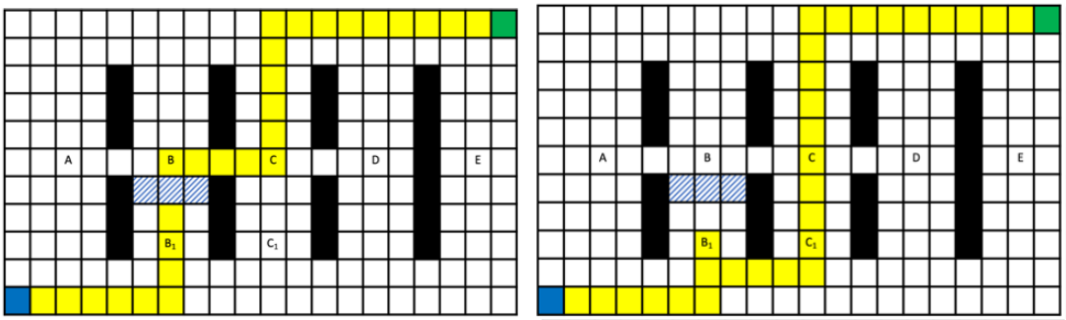
$$\pi : \mathbf{X} \rightarrow \mathbf{U} \tag{3}$$

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

Let us assume that that the robot encounters the obstacle when it reaches stage K_O. For the first policy π^1_KIf the robot decides to wait, then then it adds K_1 while if the robot decides to re-plan and execute the total path length will be K_2. We let the quantity K equal to the larger value between K_1 and K_2. Now we may write the policy for the robot to wait as π^1_K and the policy for re-planning as π^2_K. These policies are padded correspondingly to produce actions $\mathbf{u}^1_K$ and $\mathbf{u}^1_K$ that are padded with zero-cost state preserving actions.

To see which policy is better on average, we consider the expected value of the cost function for both cases. The case when policy π_K^1 is chosen is characterized by the inequality shown in Eq. (4).

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



(a) Policy 1: The path for policy involving waiting for the obstruction to move π_K^1 and continuing via aisle B

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

We can see from Fig. 1 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.

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

From Fig. 2b which shows the re-planned path, we can also see that the cost of this path is equal to the expression in Eq. (6)

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

Substituting the expressions in Eq. (5) and Eq. (6) into Eq. (4). We obtain the inequality shown in Eq. (7)

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

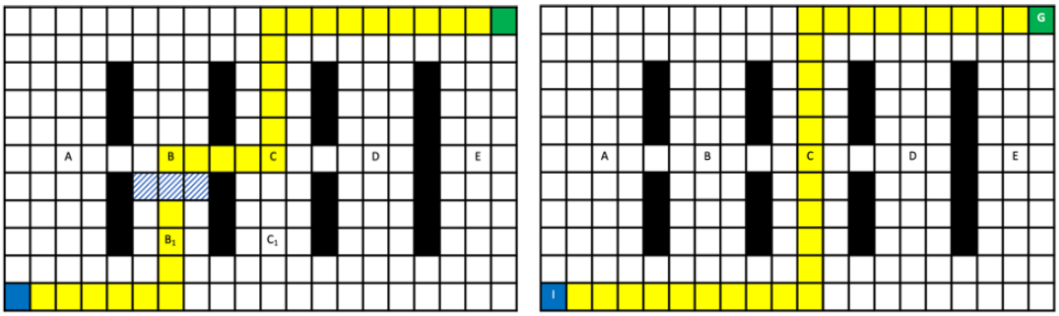
The quantity $\mathbb{E}\left[T\right]$ is the expected value for the time the robot has to wait for the obstacle to clear. As we know the distribution that the variable is sampled from we can compute the expected value. The expected value for the time taken is given by the expression found in Eq. (8)

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

Substituting the expression from Eq. (8) into Eq. (7) we obtain the inequality in Eq. (9). The right-hand side of the inequality in Eq. (9) represents 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. The inequality also takes into consideration the constraint that $\lambda_B > 0$ and negates the solution when $\lambda_B < 0$.

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

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.

Instead of reacting to the obstacle as the robot observes it, the robot may also make a decision before it starts to move. The robot has knowledge of where the obstacles will be and the probability distribution for the wait time. Depending on the probabilities involved, the robot may decide to avoid the obstacle altogether instead of going the route with the obstacle and wait. In the case of the warehouse example with 5 aisles, and one obstacle in aisle B, the two policies are to choose to travel down aisle B and wait if an obstacle is present (illustrated in Fig. 3a) or to avoid aisle B and plan directly along aisle C (illustrated

in Fig. 3b). By avoiding travelling down aisle B before replanning the robot avoids traversing the extra distance to the cell marked B_1 which makes this approach favorable.

Similar to the approach in §1.1, we may denote the policy that goes down aisle B as π_K^1 and the policy that goes down aisle C as π_K^2. We also consider the policy associated with choosing to go down aisle B and then replanning down aisle C as soon as the obstacle is observed. This policy will be represented as π_K^3. Once again these are padded with zero-cost state preserving actions so as to be of the same dimension.

Fig. 3 depicts the two possible paths that represent the two former policies. The three different policies and their costs are as follows:

- 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).

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

- Policy π_K^3 : Choose to go down aisle B and then replan down aisle C as soon as the obstacle is observed. The cost of this policy is given in Eq. (12)

$$L(\pi_k^2) = L_{IB_1} + L_{B_1C} + L_{CG} \tag{12}$$

Comparing the costs for policies π_K^2 and π_K^3 (given by Eq. (11) and Eq. (12) respectively), we can see that they only differ by only a few terms. Whereas π_K^2 goes directly down aisle C, π_K^3 first attempts to go down aisle B. This results in π_K^2 having the cost L_{IC} to get to aisle C while π_K^2 has the cost $L_{IB_1} + L_{B_1C}$ to get to aisle C. As L_{IC} is the cost of the best path, $L_{IC} \leq L_{IB_1} + L_{B_1C}$. Therefore we may ignore policy π_K^3 as the robot will always pick policy π_K^2 over policy π_K^3.

As we are concerned with the average case scenario we compare the expected value of the loss function just as we did in section 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. (13).

$$\mathbb{E}\left[L\left(\pi_K^2\right)\right] \leq \mathbb{E}\left[L\left(\pi_K^1\right)\right] \tag{13}$$

Then, substituting the loss functions from Eq. (10) and (11) into the inequality in Eq. (13), we obtain the result shown in Eq. (14).

$$\begin{aligned} \mathbb{E}[L_{IC} + L_{CG}] &\leq \mathbb{E}[L_{IB_1} + TL_W + L_{B_1B} + L_{BC} + L_{CG}] \\ \mathbb{E}[T]L_W &\geq L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC} \\ \mathbb{E}[T] &\geq \frac{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}}{L_W} \end{aligned} \tag{14}$$

By substituting the expected value for T found in Eq. (8) we obtain a constraint for λ_B as shown in Eq. (15). The right hand side of the final inequality in Eq. (15) represents the largest value for λ_B for which the robot will decide to go directly down aisle C avoiding aisle B. Once again the sultions for $\lambda_B < 0$ have been ignored.

$$\begin{aligned} \frac{1}{\lambda_B} &\geq \frac{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}}{L_W} \\ \lambda_B &\leq \frac{L_W}{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}} \end{aligned} \tag{15}$$

1.3 Considering the Probability of the Obstacle Being Present

In reality, the obstacle would not be present there all the time. This can be taken into account using a probability, say p_B associated with the scenario that the obstacle is present[1]. The mean wait time in the case the obstacle is present has been derived in Eq. (8). In the case the obstacle isn't present, the robot does not have wait and therefore the mean wait time is 0. By taking a sum of these 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. (16):

$$\begin{aligned} \mathbb{E}_B[T] &= p_B \cdot \mathbb{E}[T] + (1 - p_B) \cdot (0) \\ &= p_B \cdot \frac{1}{\lambda_B} = \frac{p_B}{\lambda_B} \end{aligned} \tag{16}$$

where, $\mathbb{E}[T]$ is expected time to wait given the obstacle in aisle B exist and it is given by Eq. 7
Given Eq. (16) we may reconsider Eq. (14) and substitute this new value for the expected wait time:

$$\begin{aligned} \mathbb{E}_B[T] &\geq \frac{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}}{L_W} \\ \frac{p_B}{\lambda_B} &\geq \frac{L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}}{L_W} \end{aligned} \tag{17}$$

Assuming that the rate parameter λ_B is constant, we may arrive at the equation shown in Eq. (18).

$$p_B \geq \frac{\lambda_B \left(L_{IC} - L_{IB_1} - L_{B_1B} - L_{BC}\right)}{L_W} \tag{18}$$

It should be noted however that this was the inequality to choose the policy to plan straight through aisle C. Therefore the R.H.S of the inequality in Eq. (18) represents the value for p_B below which the robot will attempt to drive down aisle B first.

1.4 Considering Multiple Obstacles

Let us now assume that both aisles B and C have obstacles on them. The wait time for the obstacle in aisle B to clear is sampled from the distribution outlined in Eq. (1). The probability of the obstacle in aisle B being present is once again given as p_B. Similarly the obstacle in aisle C has an identical wait time distribution as the obstacle in aisle B but with a characterizing parameter of λ_C. The probability of this obstacle being present is given by the probability p_C. Following this, the expected wait times for obstacle B and obstacle C are as follows:

$$\begin{aligned} \mathbb{E}_B[T] &= \frac{p_B}{\lambda_B} \\ \mathbb{E}_C[T] &= \frac{p_C}{\lambda_C} \end{aligned} \tag{19}$$

The robot has the same start and goal cells as described in §1.1. Given that the robot chooses a policy at the beginning, there are 6 different key scenarios that are worth considering:

- 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:

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

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

- 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 obstacles 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:

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

We ignore the case 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)

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

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:

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

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. Similarly the robot will always pick π_K^6 over π_K^3 and π_K^5. Therefore we can ignore both policies π_K^3 and π_K^5. We have therefore simplified the inequality in Eq. (26) as follows:

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

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 the $L\left(\pi_K^6\right)$ for policy π_K^6 to be chosen is either that given by Eq. (29) or by Eq. (29) whichever one is smaller. The inequalities cannot be merged 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 following inequality which will also hold:

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

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 $A < B + C + D$. It should be noted that this is not 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.

2 Implementation of the System in ROS

2.1 Planning Path via an Aisle

Figure 4: 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.

This section discusses the implementation of the principles discussed in §1. The simulations will be carried out using the STDR Simulator running on Robotics Operating System (ROS). The nodes and all supporting scripts will be coded in python. The map in Fig. 4 shows the map of the environment that will be 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. (4) implies that there are 4 temporary obstacles on

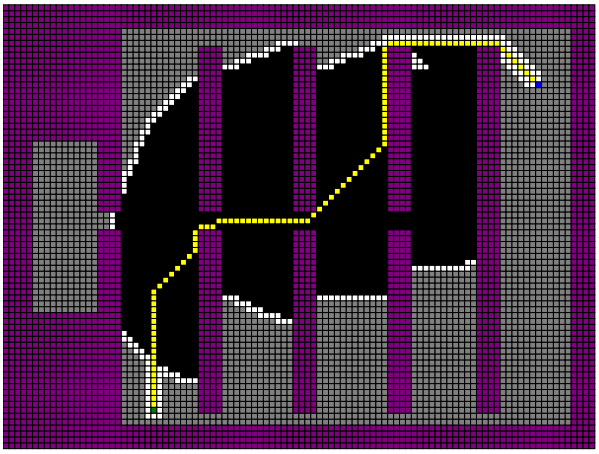
the map. However only one of them will be active throughout the simulations carried out in this section. This is the obstacle in aisle B.

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:

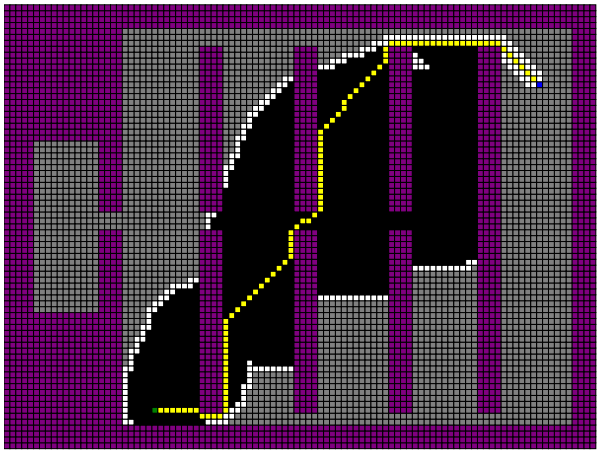
- `startCellCoords` : The coordinates of the start cell in the search grid (different from the actual coordinates of the cell on the map)
- `goalCellCoords` : The coordinates of the goal cell in the search grid (different from the actual coordinates of the cell on the map)
- `aisle` : The aisle down 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. 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.

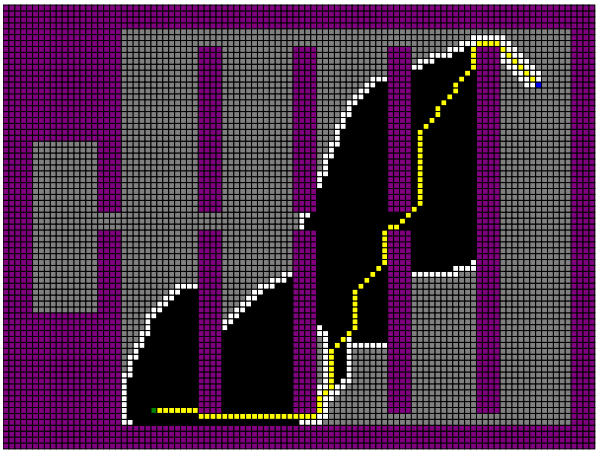
For the plotting of search grids the two separate search grids from the two executions of the search algorithm need to be merged. This was achieved by the addition of the `leftMergeGrid` method in the `SearchGrid` class. This method acts upon a search grid and takes the cell grid of another search grid 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. 5.
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` 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 a set maximum, the robot stop waiting and replans via another aisle chosen by the `chooseAisle` method. This maximum is defined by the `maxWaitTime` attribute of the `ReactivePlannerController` class which in turn is read from the ROS parameter `max_wait_time`.



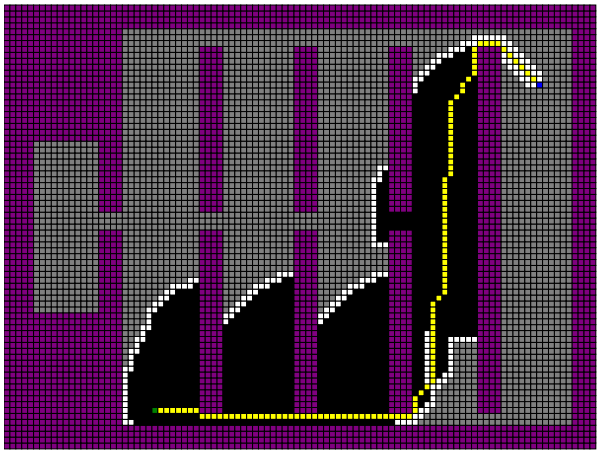
(a) Search grid for a path planned via Aisle A



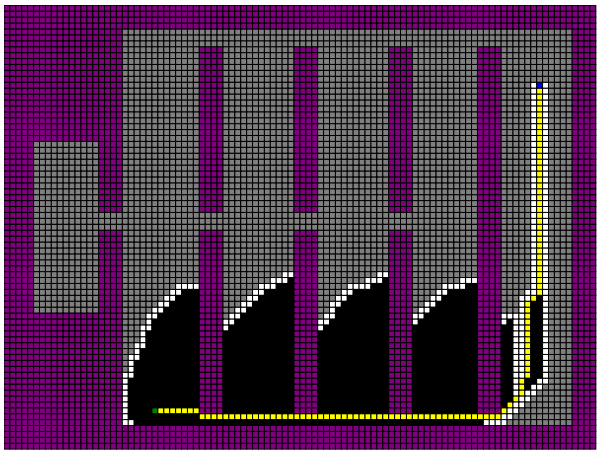
(b) Search grid for a path planned via Aisle B



(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 5: 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.

Once a robot encounters an obstacle it must decide if it is more cost efficient to replan 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 replan. To make a decision on which policy is better, the expected cost of both 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. To 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. However 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.

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

References

Appendices