RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN
INSTITUT FÜR INFORMATIK I

Elmar Langetepe

Online Motion Planning

MA INF 1314

Summersemester 2016
Manuscript: Elmar Langetepe

Chapter 1

Labyrinths, grids and graphs

In this section we first concentrate on discrete environments based on grid structures. For the grid structure we consider an agent that can move from one cell to a neighbouring cell with unit cost. We start with the task of searching for a goal in a very special grid environment. After that we ask for visiting all cells, which means that we would like to explore the environment. For this task the grid environment is only partially known, by a touch sensor the agent can only detect the neighbouring cells. The agent can build a map. Exploration and Searching are closely related. If we are searching for an unknown goal, it is clear that in the worst-case the whole environment has to be explored. The main difference is the performance of these *online* tasks. As a comparison measure we compare the length of the agent's path to the length of the optimal path under full information. Thus, in the case of searching for a goal, the comparison measure is the shortest path to the goal.

At the end of the section we turn over to the exploration task in general graphs under different additional conditions.

1.1 Shannons Mouse Algorithm

Historically the first online motion planning algorithm for an autonomous agent was designed by Claude Shannon [Sha52, Sha93] in 1950. He considered a 5×5 cellular labyrinth, the inner walls of the labyrinth could be placed around arbitrary cells. In principle, he constructed a labyrinth based on a grid environment; see Figure 1.1.

The task of his electronical mouse was to find a target, i.e. the cheese, located on one of the fields of the grid. The target and the start of the mouse were located in the same *connected component* of the *grid labyrinth*. The electronical mouse was able to move from one cell to a neighbouring cell. Additionally, it could (electronically) mark any cell by a label N, E, S, W which indicates in which direction the mouse left the cell at the last visit. This label is updated after leaving the cell. With theses abilities the following algorithm was designed.

Algorithmus 1.1 Shannons Maus

- Initialize any cell by the label N for 'North'.
- While the goal has not been found:
  starting from the label direction, search for the first cell in clockwise order that can be visited. Change the label to the corresponding direction and move to this neighbouring cell.

Sutherland [Sut69] has shown that:

Theorem 1.1 *Shannon's Algorithms (Algorithmus 1.1) is correct. For any labyrinth, any starting and any goal the agent will find the goal, if a path from the start to the goal exists.*

Figure 1.1: Shannons original mouse labyrinth.

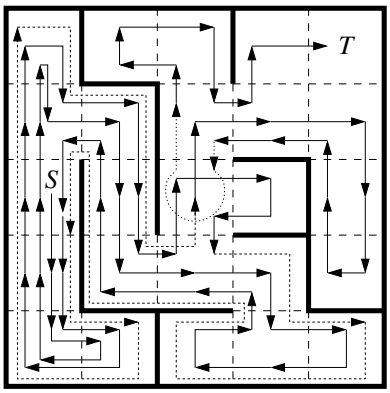


Figure 1.2: An example of the execution of Shannons Algorithm.

Proof. We omit the goal and show that any cell in the connected component of the start will be visited infinitely often. □

Exercise 1 *Formalize the above proof sketch!*

As shown in Figure 1.2 the path of Shannons Mouse is not very efficient.

1.2 Intuitive connection of labyrinths, grids and graphs

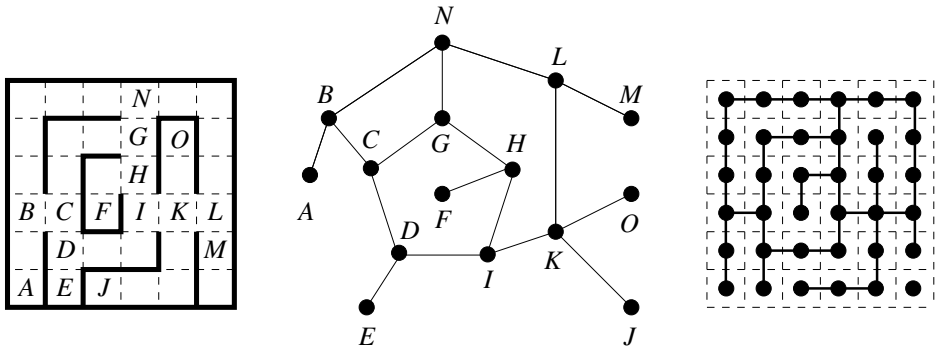
For a human a labyrinth consists of corridors and connection points. In this sense the environment for Shannons task can be considered to be a labyrinth. Obviously any such labyrinth can be modeled by a planar graph.[1] More precisely the environment for Shannons task is a grid graph. Figure 1.3 shows the corresponding intuitive interpretations.

For any intuitive labyrinth there is a labyrinth-graph. On the other hand for any planar graph we can build some sort of labyrinth. This is not true for general graphs. For example the complete graph K_5 has no planar representation and therefore a correspondance to a labyrinth does not exist.

1.3 A lower bound for online graph exploration

We consider the following model. Assume that a graph $G = (V,E)$ is given. If the agent is located on a vertex it detects all neighbouring vertices. Let us assume that moving along an edge can be done with

[1] A graph, that has an intersection free representation in the plane.



(i) Labyrinth (ii) Labyrinthgraph (iii) Gittergraph

Figure 1.3: Labyrinth, labyrinth-graph and gridgraph.

unit cost. The task is to visit all edges and vertices and return to the start. The agent has the ability of building a map. If we apply a DFS (depth first search) for the edges we will move along any edges twice. DFS can run online. The best offline strategy has to visit any edge at least once. In this sense DFS is a 2-approximation.

The comparison and approximation between online and offline is represented by the following concept. A strategy that runs under incomplete information is denoted as an **Online–Strategyn**. On the other hand an **Offline–Strategyn** solves the same task with full information. In the above example the offline strategy is the shortest round trip that visits all edges of the graph.

The performance measure for Online-Algorithms is the so-called *competitive ratio*.

Definition 1.2 (Sleator, Tarjan, 1995)
Let Π be a problem class and S be a strategy, that solves any instance $P \in \Pi$.
Let $K_S(P)$ be the cost of S for solving P.
Let $K_{opt}(P)$ be the cost of the optimal solution for P.

The strategy S is denoted to be c–competitive, if there are fixed constants $c, \alpha > 0$, so that for all $P \in \Pi$

$$K_S(P) \leq c \cdot K_{opt}(P) + \alpha$$

holds.

The additive constant α is often used for starting situations. For example if we are searching for a goal and have only two unknown options, the goal might be very close to the start, the unsuccessful step will lead to an arbitrarily large competitive ratio. This is not intended. Sometimes we can omit the additive constant, if we have additional assumptions. For example we can assume that the goal is at least distance 1 away from the start.

As already mentioned DFS on the edges visits any edge at most twice. There are graphs where the optimal offline solution also has to visit any edge twice. For such examples DFS is optimal with ratio 1. Now we are searching for a lower bound for the competitive ratio. That is, we would like to construct example such that any possible online strategy fails within a ratio of 2.

Theorem 1.3 *(Icking, Kamphans, Klein, Langetepe, 2000)*
For the online-exploration of a graph $G = (V, E)$ for visiting all edges and vertices of G there is always an arbitrarily large example such that any online strategy visits roughly twice as much edges in comparison to the optimal offline strategy. DFS always visit no more than twice as much edges against the optimum. *[IKKL00]*

Proof. The second part is clear because DFS visits exactly any edge twice. Any optimal strategy has to visit at least the edges.

The robot should explore a gridgraph and starts in a vertex s. Finally, the agent has to return to s. We construct an *open* corridor and offer two directions for the agent. At some moment in time the agent has explored ℓ new vertices in the corridor. If this happens we let construct a conjunction at one end s' of the corridor. At this bifurcation two open corridors are build up which *run* back into the direction of s. If the agent proceeds one of the following events will happen.

1. The agent goes back to s.
2. The agent has visited more than $\ell+1$ edges in one of the new corridors.

Let ℓ_1 denote the length of the part of the starting open corridor into the opposite direction of s'. Let ℓ_2 and ℓ_3 denote the length of the second and third open corridor.

We analyse the edge visits $|S_{ROB}|$ that an arbitrary strategy S_{ROB} has done so far.

1. $|S_{ROB}| \geq 2\ell_1 + (\ell - \ell_1) + 2\ell_2 + 2\ell_3 + (\ell - \ell_1) = 2(\ell + \ell_2 + \ell_3)$, see Figure 1.4. Now we close the corridors at the open ends. From now on the agent still requires $|S_{OPT}| = 2(\ell + \ell_2 + \ell_3) + 6$ edge visits, where S_{OPT} is the optimal strategy if the situation was known from the beginning. Thus we have: $|S_{ROB}| \geq 2|S_{OPT}| - 6$.

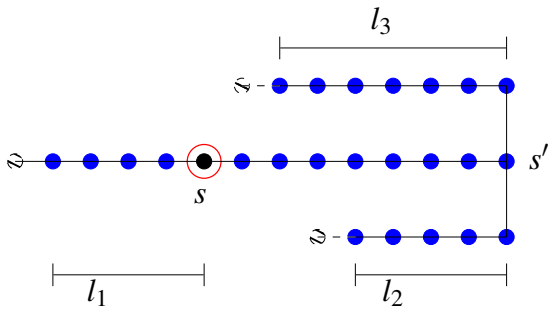


Figure 1.4: The agent return to s

2. W.l.o.g. the agent has explored $\ell+1$-ten vertices in corridor 3. We have $|S_{ROB}| \geq 2\ell_1 + (\ell - \ell_1) + 2\ell_2 + (\ell+1)$. We connect corridor 3 with corridor 1(see Figure 1.5) and close corridor 2. The agent still requires $\ell + 1 + 2(\ell_2 + 1) + (\ell - \ell_1)$ edge visits; in total at least $4\ell + 4\ell_2 + 4 = 4(\ell + \ell_2) + 4$ edge visits. From $|S_{OPT}| = 2(\ell+1) + 2(\ell_2+1) = 2(\ell+\ell_2) + 4$ we conclude $|S_{ROB}| \geq 2|S_{OPT}| - 4 > 2|S_{OPT}| - 6$.

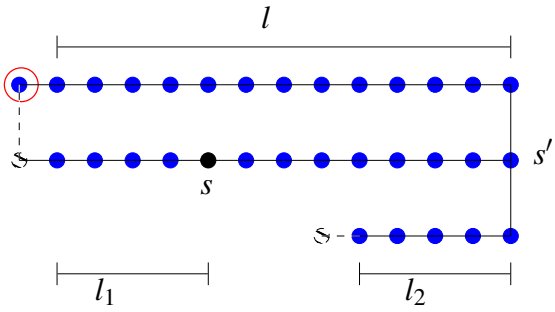


Figure 1.5: The agent has visited $\ell+1$ vertices in corridor 3.

We have $|S_{ROB}|/|S_{OPT}| \geq 2 - 6/|S_{OPT}|$. We also have $|S_{OPT}| \geq 2(\ell+1)$ and conclude $2 - 6/|S_{OPT}| > 2 - 6/2\ell = 2 - 3/\ell$. For arbitrary $\delta > 0$ we choose $\ell = \lceil 3/\delta \rceil$ and conclude $|S_{ROB}|/|S_{OPT}| > 2 - \delta$. □

Remark 1.4 *There are always examples so that the optimal exploration tour visits any edge twice.*

Corollary 1.5 *DFS for the Online-Edge-Exploration of general graphs is 2–competitive and optimal.*

Exercise 2 *Show that the same competitive ratio holds, if the return to the starting point is not required.*

Exercise 3 *Consider the problem of exploring the vertices (not the edges) of a graph. If the agent is located at a vertex it detects the outgoing edges but along non-visited edges it is not clear which vertex lies on the opposite side. Does DFS applied on the vertices result in a 2-approximation?*