Leaving an Unknown Maze with One-Way Roads

Bernd Brüggemann* Tom Kamphans* Elmar Langetepe*

Abstract

We consider the problem of escaping from an unknown polygonal maze under limited resources. The maze may have passages that can be traversed in only one direction. It is well-known that in a setting without 'one-way roads', the Pledge algorithm always finds a path out of an unknown maze provided that such a path exists. We extend the Pledge algorithm for our type of environments and show the correctness of our solutions.

Apart from the maze-leaving application, we introduce a new type of scenes that combines the advantages of both polygonal scenes (i.e., modelling the geometric shape of the environment) and directed graphs (i.e., modelling the connectivity of several parts in the environment). This type might be interesting to consider in other motion-planning tasks such as exploration and search.

Keywords: Online algorithms, motion planning, autonomous robots, Pledge algorithm, one-way roads.

1 Introduction

Imagine you want to leave the old town of a large city. The city is surrounded by some ring roads. As soon as you reach this ring, there are traffic signs that lead you to your destination, but you have no clue how to get to the ring roads. To make things worse, there are many one-way roads in the old town [8].

Usually, we model environments like this by *polygonal scenes*; see, for example, [12, 14, 3]. If we have limited ressources and, particularly, cannot build a map of the environment, the task of leaving an unknown maze can be solved using the well-known Pledge algorithm, see Algorithm 1.[1] The Pledge algorithm assumes that the searcher is able to recognize and follow a wall in a specified direction while counting the *turning angles* (w.l.o.g. we assume that the searcher uses the *left-hand rule*; that is, the searcher keeps the obstacle boundary on its left side). We assume that the searcher has no vision and knows, when it leaved the maze.

It was shown by Abelson and diSessa [1] and Hemmerling [7] that a searcher will escape from a polygonal maze using the Pledge algorithm, provided that there is such a solution, and provided that the agent is error free. For sufficient conditions on the searcher's errors to ensure a successful application of the Pledge algorithm see [9, 8].

Algorithm 1: Pledge [1]

$\varphi := 0$.
REPEAT
  REPEAT
    Move in direction φ in the free space.
  UNTIL Searcher hits an obstacle.
  REPEAT
    Follow the wall using the *left-hand rule*.
    Count the overall turning angle in φ.
  UNTIL Angle counter φ is equal to 0.
UNTIL Searcher is outside the maze.

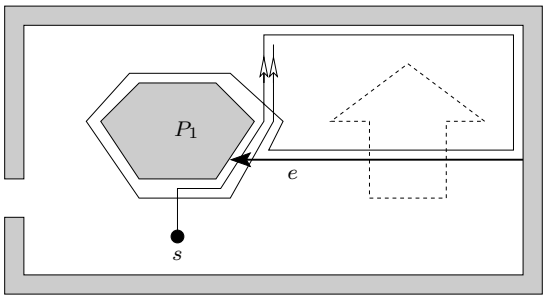

Figure 1: Applying the Pledge algorithm to environments with one-way roads does not work.

We consider the case that there are one-way roads in the surrounding. We can model this problem by adding directed edges between obstacle boundaries. The searcher is allowed to cross these edges only from the left to the right—seen in the direction of the edge—but never the opposite way. It is easy to see that we cannot simply apply the Pledge algorithm considering one-way roads as obstacle edges if we encounter them from the wrong side and otherwise pass them. Figure 1 shows an example with one one-way road e. A searcher starting in s hits the obstacle P_1, passes e, and leaves P_1. Following the second obstacle, the searcher meets the exit side of e. Thus, it follows e and circles P_1 again. The angle counter gets zero in the same vertex of P_1 as in the first visit and the searcher is trapped in an endless loop. Therefore, the

*University of Bonn, Institute of Computer Science I, 53117 Bonn, Germany.

[1] See also [1, 7, 10]. For an implementation of the Pledge algorithm see [6].

simple consideration of one-way roads does not work and we have to use a more sophisticated strategy.

2 Preliminaries

We are given a scene, $\mathcal{M} = (\mathcal{P}, \mathcal{E})$, where $\mathcal{P}$ is a set of simple polygons and $\mathcal{E}$ is a set of *directed* edges, whose start- and endpoints lie on the boundary of a polygon. We call the left side of an edge $e \in \mathcal{E}$ the *entrance* of the one-way road marked by e, the right side the *exit*. The searcher is allowed to pass e only from the entrance side; passing e from the exit side is forbidden. We assume that the searcher perceives exits as walls. Nevertheless, we assume that the searcher does not have sufficient memory to store a map of the whole scene. We allow space only in $|\mathcal{E}|$, not in $|\mathcal{P}|$.

We consider two different models: First, the searcher is not able to distinguish entrances; that is, when meeting an entrance, the searcher is not able to determine whether this entrance is met for the first time or has been met before. In the second setting we assume that the search is able to distinguish entrances. Either every entrance has a unique identifier that the searcher can read or the searcher is able to mark a discovered entrance.

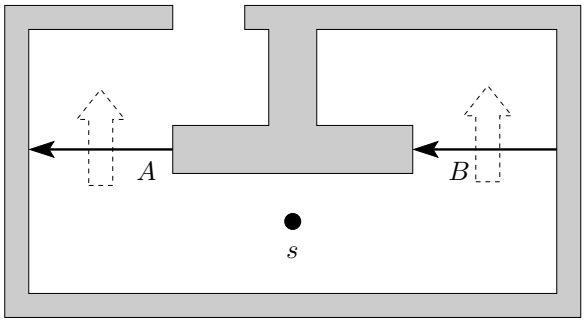

Figure 2: An unfair maze: The searcher cannot tell whether A or B leads to the exit, and it is trapped if it chooses to pass B.

Further, we assume that for every point in the free space, $\mathcal{C}_{\text{free}} := \mathbb{R}^2 \setminus \bigcup_{P \in \mathcal{P}} \mathring{P}$, there exists a path to an exit; we call such an environment a *fair* maze. In an unfair maze, the searcher may get stuck. See, for example, Figure 2: Starting in s, the searcher cannot determine whether the exit is behind one-way road A or B. Once the searcher passed the wrong road, B, it is trapped!

Definition 1 *Given a scene, $\mathcal{M} = (\mathcal{P}, \mathcal{E})$, we can consider every edge in $\mathcal{E}$ as a wall. Now, the free space, $\mathcal{C}_{\text{free}}$, divides into several path-connected components.*[2] *We call these components the* regions *of $\mathcal{M}$.*

[2] A set $S \subseteq \mathbb{R}^2$ is path connected, if for every $a, b \in S$ there is a path from a to b that is completely inside S.

3 Leaving a Maze with One-Way Roads

The difference between usual polygonal scenes and our type of environments is that we have edges that mark the entrance to a one-way road. Now, when the searcher reaches an entrance, it has the choice to enter the one-way road or to consider the entrance edge as a wall and follow the edge.

It is easy to see that any strategy with periodic choices (e.g., 'enter every second one-way road' or 'enter a one-way road every second time that its entrance is met') fails: For such a strategy, we can construct a maze where the given periodic choice ends up in an endless loop. However, any fair maze is solvable:

Lemma 1 *For every fair maze, $\mathcal{M} = (\mathcal{P}, \mathcal{E})$, there is a function $\beta : \{1, \ldots, |\mathcal{E}|\} \longrightarrow \{\text{enter}, \text{bypass}\}$ such that a searcher using the Pledge algorithm can leave $\mathcal{M}$ if it enters a one-way road, e_i, iff $\beta(i) =$ enter.*

Proof. We start with $\beta(i) =$ bypass $\forall i$. In every maze there exists a region, R_1, from which the searcher using the Pledge algorithm can escape without crossing a one-way road. Remark that R_1 is the only unbounded region in $\mathcal{M}$. Now, we remove every obstacle and one-way road in R_1 and proceed recursively until we removed every obstacle. For a given start point, there is a sequence $R_k, \ldots, R_1$ of regions that the searcher has to pass to move from s to R_1. Now, we define $\beta(i) :=$ enter for every e_i that leads from a region R_j to R_{j-1}, $1 < j \leq k$. □

3.1 Indistinguishable One-Way Roads

In this section, we assume that the searcher is not able to distinguish one-way roads. That is, if the searcher meets an entrance it cannot tell whether this is a new entrance or one that has been met before. Algorithm 2 solves the problem in this setting: We store a control word $w \in \{\text{'r'}, \text{'p'}, \text{'b'}\}^*$ (see Algorithm 2) where every character determines the searcher's behavior when meeting an entrance. We evaluate this word character by character and generate the next word in lexicographical order (basically, we add 1 in $\mathbb{Z}/3\mathbb{Z}$) when every character is evaluated. Between two entrances, the searcher moves using the Pledge algorithm.

Theorem 2 *Algorithm 2 finds the exit from every fair maze.*

Proof. We use a proof idea similar to [7]. First, we show that there is a universal control word, w_{uni}, that allows the searcher to escape from every start point. Let the searcher start in s_1 and let e_i be the first one-way road that the searcher meets. By Lemma 1, there is a control word w_i that directs the searcher to the exit. Now, let the searcher start in s_2 and met another one-way road $e_j \neq e_i$. If the searcher applys w_i

Algorithm 2: Pledge with indistinguishable one-way roads

- $w := ''\mathrm{p}''$, $i := 1$.
- Use the Pledge algorithm, until a one-way entrance is met. If the ith character in w is
  'p': enter the one-way road
  'b': do not enter
  'r': angle counter $\varphi := \varphi \bmod 2\pi$
  Increment i.
  If $i > |w|$ generate the next word and $i := 1$.
  Continue the Pledge algorithm.

starting in e_j, it either escapes or ends up on another one-way road e_k. Now, there is a word w_k that leads to the exit, and the concatenation $w_1 := w_k \circ {}'\mathrm{r}' \circ w_i$ finds the exit from two one-way roads. Note that it is necessary to 'reset' the angle counter between two control words to ensure that w_k leads to the exit for every angle-counter value that the searcher may have when it meets e_k, because two different angle-counter values may cause different paths even for the same control word.

This way we continue, until our control word, w_{uni}, finds the exit from every one-way road. Algorithm 2 enumerates all words in $\{\, {}'\mathrm{r}', {}'\mathrm{p}', {}'\mathrm{b}' \,\}^*$ and, thus, eventually finds w_{uni}. □

Needless to mention that Algorithm 2 may try a lot of words until it finds the exit and uses $O(|\mathcal{E}|^2)$ space, so this algorithm is more of theoretical value.

3.2 Distinguishable One-Way Roads

Now, we assume that the searcher is able to uniquely identify entrances to one-way roads. Clearly, this model is more powerful than the one used in the preceding section: We have the advantage of using the entrances of one-way roads as landmarks. Thus, we can uniquely identify regions by the entrances on their outer boundary. Our algorithm builds a graph of the environment with one node per region and directed edges representing the one-way roads leading from one region to another. We traverse this graph using an online strategy for the exploration of directed graphs, see Papadimitriou [13], Albers and Henzinger [2], Kwek [11], or Fleischer and Trippen [5]. The graph exploration strategy is the framework for our maze leaving strategy. Algorithm 3 is a subroutine that is called by the graph explorer when the search begins and every time the searcher has just passed a one-way road that has never been passed before.

Algorithm 3, in turn, uses basically the Pledge algorithm to explore a region. If the searcher arrives at an entrance, the Pledge algorithm is interrupted to gather some information about the region. Algorithm 3 uses a marker for every one-way roads either to store a pair (from, to) of regions or to mark the one-way road as 'unknown so far', 'open forever', or 'closed forever'. The latter is used for one-way roads that are part of the inner boundary of a region that is already visited (i.e., lead to regions that are completely surrounded by this region), see Figure 3(i). The exit of the maze cannot be inside such a region, so we never have to visit it. We use 'open forever' for one-way roads that do not define a region, see Figure 1. For every other one-way road, the decision whether or not to enter it is based on the online exploration strategy for directed graphs.

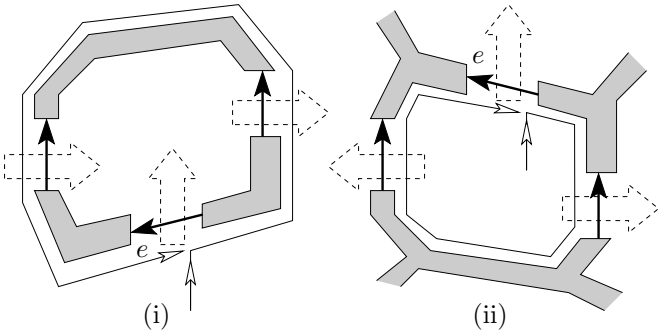

Figure 3: (i) The searcher surrounded a set of inner obstacles, (ii) the searcher moved on the outer boundary of a region.

Theorem 3 *A graph explorer that calls Algorithm 3 from start point s and after a one-way road is entered for the first time, finds the exit from every fair maze.*

Proof. The Pledge algorithm ensures that we eventually reach the outer boundary of a region that we enter either from the start point or after passing a one-way road: Inside a region we interrupt the Pledge algorithm only when we meet an entrance on the inner boundary. After surrounding this part of the inner boundary, we consider this boundary as one obstacle by 'closing' every one-way road that leads inside. Then, we continue the Pledge algorithm with the same angle-counter value as before the interruption. From the correctness of the Pledge algorithm follows that it reaches the bounding box of a maze in a setting without one-way roads; thus, in our setting the Pledge algorithm reaches the outer boundary of a region.

When we reach the first one-way road on the outer boundary of a region, we interrupt the Pledge algorithm again. We completely circle the outer boundary discovering every one-way road that leaves the current region. So we build by and by a graph that contains a node for every region that we enter and an edge for every one-way road between regions. The online graph exploration strategy ensures that this graph is completely traversed until we find the exit. □

Algorithm 3: Pledge with distinguishable one-way roads

- Use the Pledge algorithm, until an entrance to a one-way road e is met.
- If e is 'open forever' or 'closed forever' continue the Pledge algorithm.
- If the 'from'-marker of e is set, update the 'to'-marker of the most recently passed one-way road. (The searcher is inside a known region). If both markers are the same, set the one-way road to 'open forever' and remove it from the graph. Continue the graph explorer.
- If e was never met before, store the current angle-counter value, ω_e, and follow the wall using the *left-hand rule* until e is met again. On this path, store all discovered entrances in a list, ℓ, and count the turning angles. Compare the current angle-counter value to ω_e:
  - If the difference is 2π, the searcher has surrounded an inner obstacle (or a set of inner obstacles), see Figure 3(i). Mark every one-way road in ℓ as 'closed forever'. Set the angle-counter to ω_e and continue the Pledge algorithm.
  - If the difference is -2π, the searcher moved on the outer boundary of a region, see Figure 3(ii). If no entrance on the outer boundary has been met before, we have found a new region: Add a new vertex to the graph. Set the 'from' markers of the one-way roads in ℓ and the 'to'-marker of the most recently passed one-way road to this region. Continue the graph explorer.

Both the graph explorer and Algorithm 3 use $O(|\mathcal{E}|)$ space. The simple strategy by Kwek [11] uses $O(\min\{\, r|\mathcal{E}|, dr^2+|\mathcal{E}|\,\})$ edge traversals, where r is the number of vertices (regions) and d is the number of edges that have to be added to make the graph Eulerian. The more elaborated strategy by Fleischer and Trippen [5] uses $O(d^8\,|\mathcal{E}|)$ edge traversals. Altogether, the strategy presented in this section is more applicable than the one shown in the preceding section. But it assumes, that the searcher is able to distinguish one-way roads.

4 Conclusion

We introduced polygonal scenes with passages that can be traversed in only one direction, and considered the problem of leaving such a scene. This problem was solved for two different settings—indistinguishable and distinguishable one-way entrances—by combining the Pledge algorithm with other techniques that make the decision whether or not to enter a one-way road: enumerating all control words and exploring directed graphs online. Two other approaches are presented in [4] and in the forthcoming technical report. A strategy that leaves an unknown maze with one-way roads was implemented on a Khepera II robot [4].

The next step may be to ask, if and how other algorithms known for polygonal scenes such as searching, navigation, exploration/covering have to be modified in the presence of one-way roads.

References

[1] H. Abelson and A. A. diSessa. *Turtle Geometry*. MIT Press, Cambridge, 1980.

[2] S. Albers and M. Henzinger. Exploring unknown environments. In *Proc. 12th Annu. ACM Sympos. Theory Comput.*, pages 416–425, 1997.

[3] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. *SIAM J. Comput.*, 26(1):110–137, Feb. 1997.

[4] B. Brüggemann. Entkommen aus unbekannten Labyrinthen mit Einbahnstraßen. Diplomarbeit, University of Bonn, November 2006.

[5] R. Fleischer and G. Trippen. Exploring an unknown graph efficiently. In *Proc. 13th Annu. European Sympos. Algorithms*, volume 3669 of *Lecture Notes Comput. Sci.*, pages 11–22. Springer-Verlag, 2005.

[6] U. Handel, T. Kamphans, E. Langetepe, and W. Meiswinkel. Polyrobot — an environment for simulating strategies for robot navigation in polygonal scenes. Java Applet, 2002. http://www.geometrylab.de/Polyrobot/.

[7] A. Hemmerling. *Labyrinth Problems: Labyrinth-Searching Abilities of Automata*. B. G. Teubner, Leipzig, 1989.

[8] T. Kamphans. *Models and Algorithms for Online Exploration and Search*. Dissertation, University of Bonn, 2005. http://www.kamphans.de/k-maole-05.pdf.

[9] T. Kamphans and E. Langetepe. The Pledge algorithm reconsidered under errors in sensors and motion. In *Proc. of the 1th Workshop on Approximation and Online Algorithms*, volume 2909 of *Lecture Notes Comput. Sci.*, pages 165–178. Springer, 2003.

[10] R. Klein. *Algorithmische Geometrie - Grundlagen, Methoden, Anwendungen*. Springer, Heidelberg, 2nd edition, 2005.

[11] S. Kwek. On a simple depth-first search stratey for exploring unknown graphs. In *Proc. 5th Workshop Algorithms Data Struct.*, pages 345–353, 1997.

[12] V. J. Lumelsky and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[13] C. H. Papadimitriou. On the complexity of edge traversing. *J. ACM*, 23:544–554, 1976.

[14] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoret. Comput. Sci.*, 84(1):127–150, 1991.