The Thirteenth International Symposium
on Combinatorial Search (SoCS 2020)

Solving the Watchman Route Problem on a Grid with Heuristic Search[*]

Shawn Seiref
Ben Gurion Univ.
Be'er Sheva, Israel
shawn@post.bgu.ac.il

Tamir Jaffey
Ben Gurion Univ.
Be'er Sheva, Israel
tamiry@post.bgu.ac.il

Margarita Lopatin
Ben-Gurion University
Be'er Sheva, Israel
marglup@rnd-hub.com

Ariel Felner
Ben Gurion Univ.
Be'er Sheva, Israel
felner@bgu.ac.il

Introduction

In the *Watchman Route Problem* (WRP), we are given a map and the task is to (offline) find a route that sees every point in the map. WRP was proven to be NP-hard for polygons (Chin and Ntafos 1986) and in full paper we prove that it is NP-hard on grids too. Nevertheless, finding an optimal WRP route in a grid with heuristic search is our main focus.

The input is a grid-map M. The set of traversable (non obstacles) cells is denoted by $\mathcal{C}$. A cell $start \in \mathcal{C}$ is also given as input. A path $\pi = \langle s_0 = start, \dots, s_k \rangle$ is a sequence of adjacent cells starting from *start*. In a *watchman path* π, for every cell $c \in \mathcal{C}$ there is line-of-sight from at least one cell $s_i \in \pi$. Our task is to find an optimal (= minimal cost) watchman path.

The *line-of-sight* (LOS) function determines whether any given two cells can visually see each other and it can be any arbitrary function. In this paper we use **Bresenham LOS (*BresLos*):** a function commonly used in computer graphics, video games and bitmap pictures (Bresenham 1965). It is perhaps the most suitable LOS function that discretizes real-world continuous domains and simulates a continuous field of view. Figure 1(left) shows *BresLos* for the Green cell.

WRP is different than SLAM problems where an autonomous moving agent needs to explore the environment and build a map while simultaneously locate itself in the map (Taketomi, Uchiyama, and Ikeda 2017). A reminiscent offline problem is the *Art Gallery Problem* (AGP) (Garey and Johnson 1979) where the task is to find the minimal set of points S (not a tour as in WRP) on the map such that all points in the map can be seen by at least one point $s \in S$.

WRP as a Search Problem

We next formulate WRP as a search problem and define its corresponding search tree.

Node: A node is a pair $\langle location, seen \rangle$ where $location$ is a cell (current location of the agent) and *seen* is a list of cells (all the cells that have been seen so far by the agent). The complement of *seen* is the *unseen* list; their union is the entire set of cells (*seen* $\cup$ *unseen* $= \mathcal{C}$).

Root Node: $Root$ is a node such that $Root.location = start$ and $Root.Seen = LOS(start)$.

Expansion: Expanding node $S = \langle location, seen \rangle$ is to perform all legal movements on $S.location$. For each child S' of S, $S'.location$ is the neighboring cell of $S.location$ derived from the movement. $S'.seen$ is first inherited from the parent $S.seen$. Then, we add to $S'.seen$ all the cells that are now seen from $S'.location$ and were not seen before (i.e., $S'.seen = S.seen \cup LOS(S'.location)$). The cost of the edge from S to S' is the cost of the movement action.

Goal Node: $Goal.location$ may be any cell in $\mathcal{C}$ such that $Goal.seen = \mathcal{C}$.

Every node S in this search tree is associated with a path $\pi = \langle s_0 = start, ..., s_k = S.location \rangle$ which is determined by the branch of the search tree associated with S. $S.seen$ includes all the cells that have LOS to at least one of the locations in the path associated S. The cost of node S in the tree is the sum of the costs of applying the operators from $Root$ to S. We use *Open* to denote the Open list of the A* search that is activated on this search tree.

A* relies on an admissible heuristic to return optimal solutions. We introduce such heuristics next.

Singleton Heuristic

Our first heuristic is based on the idea that in order to solve WRP the watchman agent must see each of the cells from $S.unseen$. Thus, for each cell $p \in S.unseen$ (denoted as the *pivot* p) we define its *singleton heuristic* to be the minimal distance from $S.location$ to a cell $q \in LOS(p)$. Formally, given a pivot cell $p \in S.unseen$:

$$h_p(S) = \min_{q \in LOS(p)} d(S.location, q)$$

where $d(x, y)$ is the cost of the shortest path between cells x and y. For every $p \in S.unseen$, $h_p(S)$ is admissible because the agent will surely travel to some cell that has LOS to p and $h_p(S)$ takes the minimum among all those cells.

Every possible pivot has its own singleton heuristic. Therefore, we can take the maximum of each of these heuristics in order to maintain admissibility (Holte et al. 2006). In our experiments below we took the extreme case of maximizing over *all* cells in the *unseen* list as pivots.

[*]This paper is an extended abstract of a paper with the same title that appears at ICAPS-2020.

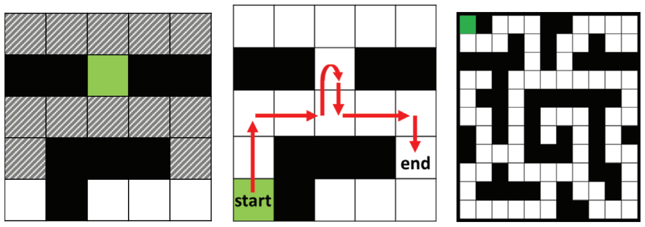

Figure 1: BresLos view, Optimal path and a 11x11 maze

Graph Heuristics

Our next two heuristics are based on a graph called the *Disjoint LOS Graph* ($G_{DLS} = (V, E)$). G_{DLS} is abstracted from the grid map M for every node S in *Open*. We say that two cells are $LOS\text{-}disjoint$ if there is no cell that they both see. We say that a set of cells P is $LOS\text{-}disjoint$ if every two cells in P are $LOS\text{-}disjoint$.

G_{DLS} is built by identifying a set of $LOS\text{-}disjoint$ cells $P \in S.unseen$. Each cell in P is called a *pivot* and is added as a node to G_{DLS}. For each pivot p we also add all cells in $LOS(p)$ as *watcher vertices* to the graph. The edges between pivot p and its *Watchers* has weight of 0, the *Watchers* are connected with *Watchers* of different *Pivots* with the weight of the true distance cost between them. The current location of the agent $S.location$ and its *Watchers* are also similarly added to G_{DLS}. We are interested in the minimum-cost Hamiltonian path from $S.location$ that visits all other vertices in G_{DLS}. This is a lower bound on the path that will see all vertices in $S.unseen$.

MST heuristic The *Minimum Spanning Tree* (MST) of a graph is the spanning tree with the minimal sum of edge costs. The MST heuristic computes a MST of G_{DLS}. MST of a graph is a lower bound on a Hamiltonian path through that graph which ensures admissibility of the MST heuristic.

TSP heuristic A more informed but more expensive heuristic is a modification of the *Traveling Salesman Problem* (TSP) tour, which calculates a minimal cycle in complete. We slightly modified a TSP solver to find the minimal path that starts at location c and passes in all pivots of G_{DLS}.

Reducing the Size of the Search Tree

Trivially, when node S is expanded, then new nodes are generated for all the cells that are neighbours of $S.location$. However, based on G_{DLS} we can significantly reduce the size of the search tree. An optimal path must include at least one watcher for each pivot in G_{DLS}. We are interested in the first such watcher. The optimal watchman path will include one of them as the first such watcher. To have a complete search (and not lose any possible path) we add *all* of them as neighbours in the search tree. Formally, when expanding a node S, we generate one child C for each edge in G_{DLS} that connects $S.location$ to a watcher W as follows: $C.location = W$. The cost of edge (S, C) in the search tree is set to the cost of the corresponding edge in G_{DLS}. $C.seen$ is updated to also include $LOS(p)$ for each cell $p \in \pi(S.location, W)$. This method is denoted as *Jump to Frontier expansion* (JF).

		Basic Expansion		Jump to Front	
LOS	h	**Nodes**	**Time**	**Nodes**	**Time**
BresLos OPT=57	BFS	149,450	6,345	2,316	503
	Singleton	61,959	2,639	1,175	246
	MST	921	143	93	44
	TSP	293	75	35	18

Table 1: Nodes expanded and CPU time on the 11x11 maze.

Experimental Results

We have performed extensive experiments on many maps. Indeed, each of our improvements achieved a significant reduction in performance. The baseline breadth-first search algorithm (labelled BFS) which did not have any heuristic could only solve relatively small problem instances within reasonable computing resources while the best method (TSP+JF) could solve much larger problem instances.

Table 1 shows representative experimental results on the 11x11 maze map shown in Figure 1 with a specific start state (the top left green cell). The columns give the number of nodes expanded and the CPU time (in msec) to fully solve the problem for the *basic expansion* (left) and for JF (right) for whem using *BresLos*. Rows correspond to the different algorithms. For BFS, JF outperformed basic expansion by almost a factor of two orders of magnitude. This factor is naturally smaller when the heuristics were added. However, even when JF was applied the heuristics further achieved a significant reduction over BFS. For *BresLos*, TSP+JF solved the entire problem in only 18ms.

Acknowledgments

The research was supported by Rafael Advanced Defense Systems, by Israel Science Foundation (ISF) grant #844/17 and by the Cyber grant by from the Prime Minister office.

References

Bresenham, J. E. 1965. Algorithm for computer control of a digital plotter. *IBM Systems journal* 4(1):25–30.

Chin, W.-P., and Ntafos, S. 1986. Optimum watchman routes. In *Proceedings of the second annual symposium on Computational geometry*, 24–33. ACM.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co.

Holte, R. C.; Felner, A.; Newton, J.; Meshulam, R.; and Furcy, D. 2006. Maximizing over multiple pattern databases speeds up heuristic search. *Artificial Intelligence* 170(16):1123–1136.

Taketomi, T.; Uchiyama, H.; and Ikeda, S. 2017. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications* 9(1):16.