Optimally Solving the Multiple Watchman Route Problem with Heuristic Search (Extended Abstract)

Yaakov Livne, Dor Atzmon, Shawn Skyler,
Eli Boyarski, Amir Shapiro, Ariel Felner

Ben-Gurion University of the Negev
{livnya, dorat, shawn, boyarske}@post.bgu.ac.il, {ashapiro, felner}@bgu.ac.il

Introduction and Definitions

In the *Watchman Route Problem* (WRP), the task is to find a path for a traveling agent (*watchman*) on a map such that any location on the map is seen from at least one location on the path (Chin and Ntafos 1986; Seiref et al. 2020). In WRP, the sight of the agent is modeled using a *Line-of-Sight* (LOS) function. Recently, WRP has been optimally solved on grids using heuristic search while proving that it is NP-hard (Seiref et al. 2020). Hereafter, we refer to that paper as `S20`. `S20` developed a variant of A* called WRP-A*. A simple admissible heuristic, called $h_{Singleton}$, was proposed. It uses the cost of reaching a location that sees the farthest cell. More complex admissible heuristics were based on abstraction of the grid to the *Disjoint Line-of-Sight Graph* (G_{DLS}). The best heuristic was based on a solution to a variant of the *Traveling Salesman Problem* (TSP) applied on G_{DLS}.

In this paper, we extend WRP to the case of multiple agents (watchmen) and denote this problem as *Multiple Watchman Route Problem* (MWRP). The input for MWRP is a grid map and a set of agents, where each agent has a start cell. Between two consecutive timesteps, each agent can perform a *move action* to one of its empty neighbouring cells. All agents have a LOS function. Following Yaffe, Skyler, and Felner (2021), we work only with *BresLos*. Our work can be easily adjusted to other LOS functions or to cases where agents can perform other movements on the grid, e.g., more complex moves (Rivera, Hernández, and Baier 2017).

In MWRP, the task is to find a path for each of the agents from its start cell through the grid such that all empty cells in the map will be covered by LOS from at least one cell of one of the paths. That is, if all agents follow their paths then every cell will be seen by at least one of the agents. Following `S20`, we assume that agents do not have to return to their start cells and the whereabouts of agents after all cells have been seen is of no importance.

The cost of a single path is the number of actions performed in it. A set of paths is *optimal* iff it has the minimal cost among all sets of paths, according to a given objective function for aggregating the costs of multiple paths. While in WRP the shortest path is desired, in MWRP, we consider two objective functions: (1) The sum of the costs (SOC) of all the paths of the agents, and (2) *Makespan* (MKSP) – the cost of the longest path among these paths. MWRP is applicable in WRP applications where multiple agents exist, such as in a museum with guards that need to tour the venue.

Modeling MWRP as a Search Problem

To solve MWRP with heuristic search, we first define the corresponding search tree which generalizes that of `S20`. A **Node** consists of (1) the current locations of the agents and (2) a set of cells that have already been seen by at least one of the agents. The **Root** node contains (1) the start cells of all agents and (2) all cells that are seen from these start cells. A node is a **Goal** node if all cells are seen. As the solution may contain paths of different lengths, we add a *terminate action* which prevents the agent to move and it stays in its last location. When a node is expanded, a new node is generated for each possible combination in the Cartesian product of actions (move or terminate) for the agents. We exclude the combination of all agents performing a terminate action.

MWRP-A*: A*-like Search Algorithm

Following `S20`, we propose MWRP-A*, an A*-like search algorithm for optimally solving MWRP. In MWRP-A*, each node n is associated with $g(n)$ and $h(n)$. $g(n)$ represents the cost of reaching n from the root. That is, $g(n)$ is calculated according to SOC or MKSP. $h(n)$ estimates the remaining cost for reaching a goal from n.

Singleton Heuristic. In MWRP all cells need to be seen. Thus, in $h_{Singleton}$, we first calculate the cost of seeing each unseen cell by the closest agent. Then, $h_{Singleton}$ takes the maximum value among all unseen cells as a heuristic. As each unseen cell needs to be seen, it is easy to see that $h_{Singleton}$ is admissible for both SOC and MKSP.

MTSP Heuristic. `S20` abstracted the grid into a graph and showed that a TSP solution on that graph is an admissible heuristic to WRP. We generalize their heuristic to h_{MTSP} and use the solution to the *Multiple Traveling Salesman Problem* (MTSP) (Kivelevitch, Cohen, and Kumar 2013) as an admissible heuristic to MWRP.

Combining Heuristics. Our two heuristic functions can be considered as a simple heuristic ($h_{Singleton}$) and a complex heuristic (h_{MTSP}). As both are admissible, they can

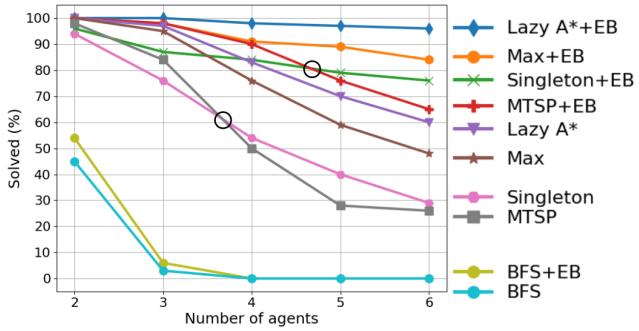

Figure 1: Success rate: 11 × 11 grid from S20 for MKSP

be admissibly combined. Thus, we consider **(1)** ***Max***, which calculates both heuristic functions for each node and takes their maximum as a heuristic, and **(2)** ***Lazy A**** (Tolpin et al. 2013), which first calculates $h_{Singleton}$ and calculates h_{MTSP} when the node is extracted from *Open*. A node is expanded if both heuristics have already been calculated.

Expanding Border Enhancement (EB). In the standard expansion method, when a node is expanded, new nodes are generated for each combination of move actions (or termination) for the agents. For some of these move actions, no new unseen cells are seen. We therefore suggest a more advanced expansion mechanism, called the *Expanding Border* mechanism (EB). EB is inspired by Jump Point Search (JPS) (Harabor et al. 2019), a framework that implements A* on grids. In EB, when expanding a node, agents may jump to farther locations than their immediate neighbors. For each non-terminated agent located at cell c, a breadth-first search (BFS) is executed from its location. If the BFS encounters a cell c' that has a LOS to some unseen cell, we consider the movement to c' as a direct move action. In EB, all such cells c' are the direct neighbors of cell c. Unlike basic expansion, with EB, agents may be in their locations at different timesteps for a given node n. Thus, these nodes do not represent a true time-space configuration. Therefore, we now directly compute the f-values by aggregating the cost-so-far (usually treated as g-values when there is no time difference) directly into the f-values.

Experimental Results

We first evaluate the following variants of MWRP-A* for minimizing MKSP: **(1)** Standard breadth-first search with no heuristic (BFS). **(2)** MWRP-A* with $h_{Singleton}$ (Singleton). **(3)** MWRP-A* with h_{MTSP} (MTSP). **(4)** The maximum between both heuristics (Max). **(5)** Lazy A* that first calculates $h_{Singleton}$ and, if needed, also calculates h_{MTSP} (Lazy A*). We use "+EB" to denote the activation of EB.

We experimented on 100 problem instances for 2–6 agents whose start cells were randomly allocated on the 11 × 11 map taken from S20. Figure 1 shows the success rate (y-axis) under a time limit of 5 minutes for different numbers of agents (x-axis) for MKSP. As expected, for all solvers, increasing the number of agents decreases the success rate. Not using any heuristic (BFS), even with EB, performed poorly. Clearly, EB increased the success rate. MTSP tends to outperform Singleton for a small number of agents but Singleton is better for more agents (the cross points are marked) due to the large overhead of calculating MTSP for many agents. Max had better success rate than both heuristics alone, and Lazy A* was always the best.

#Agents	Solved	h (+EB)	MKSP Cost	MKSP Exp.	MKSP Time	SOC Cost	SOC Exp.	SOC Time
2	94%	Singleton	30.1	6,901	38.6	52.9	21,714	47.4
		MTSP		108	16.1		387	6.6
		Max		**101**	15.5		**365**	6.8
		Lazy A*		**101**	**10.2**		366	**5.5**
3	74%	Singleton	19.1	2,124	26.9	43.6	23,731	55.3
		MTSP		51	20.3		432	9.8
		Max		**41**	19.1		**368**	9.9
		Lazy A*		**41**	**3.9**		**368**	**6.7**
4	66%	Singleton	14.9	790	26.2	36.1	17,646	57.7
		MTSP		94	50.9		632	19.7
		Max		**34**	35.6		**457**	18.0
		Lazy A*		**34**	**2.1**		**457**	**9.5**
5	61%	Singleton	12.2	493	23.4	30.5	11,009	59.4
		MTSP		73	48.0		334	16.7
		Max		**14**	26.8		234	16.8
		Lazy A*		**14**	**3.0**		**233**	**7.4**
6	47%	Singleton	10.1	299	17.8	23.7	3,664	60.5
		MTSP		99	52.9		114	12.6
		Max		**15**	34.0		**73**	12.5
		Lazy A*		18	**3.0**		**73**	**6.8**

Table 1: Results on the 11 × 11 grid for MKSP and SOC

We also experimented with minimizing SOC on the same map. Table 1 presents the results, only for our solvers with EB and excluding BFS on 2–6 agents (1st column). The percentage of instances that were solved by all solvers (MKSP and SOC) are presented in column 2. For these instances, we measured the average cost, number of expansions, and time (in sec). For each heuristic (column 3), columns 4–6 present results for minimizing MKSP and columns 7–9 for SOC.

Naturally, the average cost for SOC is higher than that of MKSP because in SOC we sum up the cost of all paths while in MKSP the cost is determined by the longest path. For minimizing MKSP, MTSP consumed larger runtime as the number of agents increased (due to its exponential overhead), while Singleton consumed shorter runtime as the number of agents increased. Hence, for more than 4 agents, Singleton outperforms MTSP, in terms of runtime. However, for minimizing SOC, MTSP always outperforms Singleton. There are two reasons for this: (1) Singleton is less accurate when minimizing SOC. (2) As shown by Kivelevitch, Cohen, and Kumar (2013), it is computationally harder for MTSP solvers that use constraint programming to minimize MKSP than minimizing SOC. Again, Lazy A* performed best. In general, there is a trade-off between the branching factor and the depth of the solution. More agents increases the branching factor but decreases the solution depth.

To summarize, the best balance between simplicity and efficiency is achieved by Singleton, which is easy to implement and achieves reasonable success rate. Lazy A* achieved the best performance, but is harder to implement.

Acknowledgments

This research was sponsored by the United States-Israel Binational Science Foundation (BSF) under grant numbers 2017692 and 2021643, and by Israel Science Foundation (ISF) under grant number 844/17.

References

Chin, W.-P.; and Ntafos, S. 1986. Optimum watchman routes. In *the Second Annual Symposium on Computational Geometry*, 24–33.

Harabor, D. D.; Uras, T.; Stuckey, P. J.; and Koenig, S. 2019. Regarding Jump Point Search and Subgoal Graphs. In *the International Joint Conference on Artificial Intelligence (IJCAI)*, 1241–1248.

Kivelevitch, E.; Cohen, K.; and Kumar, M. 2013. A Binary Programming Solution to the Min-Max Multiple-Depots, Multiple Traveling Salesman Problem. In *AIAA Infotech at Aerospace Conference*, 1–11.

Rivera, N.; Hernández, C.; and Baier, J. A. 2017. Grid Pathfinding on the 2k Neighborhoods. In *the AAAI Conference on Artificial Intelligence (AAAI)*, 891–897.

Seiref, S.; Jaffey, T.; Lopatin, M.; and Felner, A. 2020. Solving the watchman route problem on a grid with heuristic search. In *the International Conference on Automated Planning and Scheduling (ICAPS)*, 249–257.

Tolpin, D.; Beja, T.; Shimony, S. E.; Felner, A.; and Karpas, E. 2013. Toward rational deployment of multiple heuristics in A*. In *the International Joint Conference on Artificial Intelligence (IJCAI)*, 674–680.

Yaffe, T.; Skyler, S.; and Felner, A. 2021. Suboptimally Solving the Watchman Route Problem on a Grid with Heuristic Search. In *the International Symposium on Combinatorial Search (SoCS)*, 106–114.