Minimizing State Exploration While Searching Graphs with Unknown Obstacles (Extended Abstract)

Daniel Koyfman[1], **Shahaf S. Shperberg**[1], **Dor Atzmon**[2], **Ariel Felner**[1]
[1]Ben-Gurion University, Israel
[2]Bar-Ilan University, Israel
koyfdan@post.bgu.ac.il, shperbsh@bgu.ac.il, dor.atzmon@biu.ac.il, felner@bgu.ac.il

1 Introduction

The A^* algorithm (Hart, Nilsson, and Raphael 1968) optimally solves the shortest path problem by executing a best-first search, guided by a heuristic that estimates the cost to the goal. In this paper, we focus on finding a shortest path in a special type of graphs, denoted as *graphs with unknown obstacles* (GUO). In GUO, the structure of the graph is known, but some states might be blocked. For example, some junctions in a roadmap might be closed due to weather conditions, construction, or accidents. Similarly, some servers/switches/hubs may be unavailable when routing files over a network. We assume that the status of a state (blocked/free) remains fixed for the entire duration of the search and the execution.

Identifying whether a state is free or blocked in a GUO requires an *exploration* operation, which may come with a cost. For instance, exploring a location using sensors in robotic navigation might be expensive or time-consuming. Additionally, exploration is costly when privacy needs to be preserved, i.e., when an adversary can detect exploration operators. We thus differentiate between an *exploration*, a real-world, possibly very costly, operation, and an *expansion*, a computational operation done in the CPU and only incurs time overhead. Usually, analysis on A* does not differentiate between exploration and expansion, and treats exploration as part of the expansion processes (i.e., when a node is expanded, all neighboring states are generated and explored). Therefore, A* aims to speed up the search by minimizing the number of node expansions. By contrast, this paper aims to find the shortest path while minimizing the number of explorations, even at the price of increasing the number of node expansions. Note that when the exploration operator is very costly (time-consuming) compared to the time of the expansion operator, then reducing the number of explorations is a better way to minimize the CPU time to find a solution than reducing the number of expansions (as done by A*).

In this paper, we introduce *Minimize Exploration* A^* (MXA*), a two-level search algorithm capable of searching any GUO. The high level of MXA* runs A* to find an optimal path from $start$ to $goal$. Once it reaches a state for the first time, it *explores* it. Thus, information about which states are free and which are blocked is continuously being collected. The low level calculates a heuristic for the high level by finding the shortest path to $goal$ on the *currently known graph*, bypassing known blocked states while assuming that all states that are yet unexplored are free. The resulting heuristic is more informed than classic heuristics, which assume that *all* states are free. Our low-level heuristic reduces the number of nodes expanded by the high-level A* and, as a result, reduces the number of explorations. Naturally, the tradeoff is a large number of low-level expansions. Experimental results show that MXA* reduces the number of explorations by up to an order of magnitude compared to plain A* on a number of common benchmark domains.

2 Definitions and Background

A *graph with unknown obstacles* (GUO) $G = (V, E, c, EXP)$ consists of a set of states V (or vertices), a set of (directed) edges $E \subseteq V \times V$, a cost function $c(e)$ for traversing an edge $e \in E$ ($c : E \rightarrow \mathbb{R}^+$), and an exploration function $EXP : V \rightarrow \{free, blocked\}$. State $s_2 \in V$ is a neighbor of state $s_1 \in V$ if $(s_1, s_2) \in E$. A neighboring function $N(s)$ receives a state s and returns all its neighbors. Each state in V is either *free* or *blocked*, and we assume that the status of states remains fixed for the entire duration of the search and the execution. Whether a state is free or blocked is not given as input. Instead, the exploration function EXP receives a state and returns whether it is free or blocked. Naturally, if a state is blocked, then all its incident edges are also blocked. A state that EXP has yet been executed on it is referred to as *unknown*.

The input to a *GUO-pathfinding* problem consists of a GUO $G = (V, E, c, EXP)$, a start state $start \in V$, a goal state $goal \in V$, and a heuristic function $h : V \rightarrow \mathbb{R}^+$. A *valid path* between two states s_1 and s_2 is a sequence of *neighboring free states* that starts with s_1 and ends with s_2. A path's cost is the sum of the costs of its edges. $d(s_1, s_2)$ denotes the cost of the shortest valid path between s_1 and s_2. The heuristic function $h(s)$ estimates $d(s, goal)$ for any given state s. A solution to the problem is a shortest valid path between $start$ and $goal$.The problem we solve in the paper is the *minimize exploration shortest-path problem* (MXSP), where the input is a GUO-pathfinding problem, and the task is to find a shortest valid path while minimizing the number of explore operations (i.e., calls to EXP).

3 Minimize Exploration A* (MXA*)

MXA* is a two-level algorithm. Its *high level* is similar to A*, which activates the EXP operator on the GUO. After a node n', corresponding to state s', is generated, the *low-level search* is called to calculate a *dynamic* heuristic value for s', $h_D(s')$, and the f-value of n' is recalculated with respect to $h_D(s')$. The low-level search exploits the most updated knowledge on *blocked* states and performs a search strictly in memory to calculate a more informed heuristic. Let n be a newly created node in the high level. The low level is invoked to find the shortest path from $n.s$ to $goal$ that does not traverse through any state in BLOCKED. The cost of this path is returned to the high level as $h_D(s)$. Thus, the low-level searches an abstract graph where states that were already explored are treated as *free* or as *blocked* according to the outcome of their EXP action. Additionally, we make the *free-space assumption* (Koenig and Smirnov 1997) and treat all *unknown* states as *free*. The cost of the path returned by the low level is clearly a lower bound on the shortest valid path, as the returned path may include *blocked* states (currently unknown and treated as free by the low level). Also, it is clearly more informed than any static heuristic used by A*, which assumes that *all* states are free. As the high-level search progresses, new states are revealed as blocked and are inserted into BLOCKED. Therefore, when a node is selected for expansion by the algorithm, its h-value may not be up-to-date with the BLOCKED list (when its h-value was calculated, BLOCKED contained fewer states). To remedy this, when a node n is chosen for expansion by the high level, we perform another low-level search and re-calculate $h_D(n.s)$. If $g(n) + h_D(n.s) > f_{min}$, we insert n back to OPEN. Importantly, this re-calculation of the heuristic is not relevant to A* as its heuristic is static throughout the search.

When A* and MXA* expand a node, they immediately explore all its yet unexplored neighbors and insert nodes into OPEN only if the states of the nodes are free. However, this exploration can be delayed. To do so, we first treat the neighboring states of an expanded node as free. Nodes for these states are immediately inserted into OPEN. Then, when a node is chosen to be expanded, we execute EXP on its state and discard it if it turns out to be blocked. We call this approach *Lazy Exploration* (LE).

3.1 Experiments: Comparing Explorations

We empirically compared the number of explorations performed by A* and MXA*, with and without lazy expansion (LE), on two domains: 4-connected grids (with the Manhattan Distance heuristic), and 8-connected grids (with the Octile Distance heuristic). We experimented on five grid domains, representing different topologies, extracted from the *MovingAI* repository (Sturtevant 2012): Game, Maze, Random, Room, and City. Overall, we generated over 36.5k random problem instances across all five domains.

Table 1 presents the average number of explorations for each approach. The results show that MXA* consistently performed fewer explorations than A*. Additionally, enabling LE yielded a reduction in exploration for both A* and MXA*. Notably, in our experiments, MXA*+LE explored the fewest states, outperforming A* by a substantial margin ranging from 1.7 (mazes) to 5.6 (cities) in 4-connected grids and from 1.6 (mazes) to 3.3 (cities) in 8-connected grids. Figure 1 shows per-instance outcomes on 4-connected grids of the same maps and problem instances used in the above experiment (Table 1). The y-axis, presented on a logarithmic scale, shows the improvement factor between the states explored by A* and those explored by MXA*+LE for each individual instance. The instances are sorted in increasing order of the improvement factor. The depicted results highlight that the reduction in state exploration achieved by MXA*+LE over A* is exponentially distributed and varies from an improvement factor of close to 1 up to a staggering factor of up to 278. This variability shows that MXA*+LE can significantly enhance exploration in some scenarios while consistently delivering improved performance across a broad spectrum of instances.

		A*	A*+LE	MXA*	MXA*+LE
4	Game	9,094	8,766	4,143	**3,602**
	Maze	17,790	17,727	11,458	**10,347**
	Random	12,355	11,560	4,695	**3,330**
	Room	20,318	19,443	6,516	**5,195**
	City	17,569	16,661	4,133	**3,151**
8	Game	7,759	7,366	4,128	**3,536**
	Maze	9,023	8,949	6,197	**5,616**
	Random	8,981	8,047	5,585	**3,447**
	Room	13,128	12,509	6,347	**5,111**
	City	20,129	19,033	7,453	**6,149**

Table 1: Average number of explorations for A* and MXA*, without and with LE, on 4-connected and 8-connected maps.

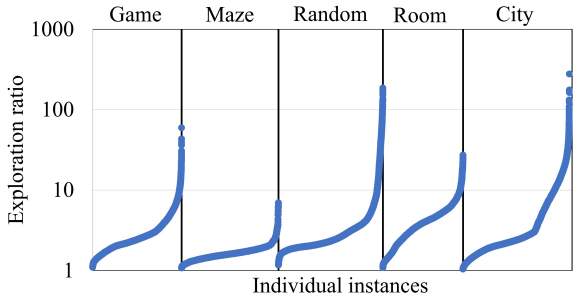

Figure 1: Per-instance exploration ratio of A* / MXA*+LE on 4-connected grids.

References

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Koenig, S.; and Smirnov, Y. 1997. Sensor-based planning with the freespace assumption. In *ICRA*, 3540–3545.

Sturtevant, N. R. 2012. Benchmarks for grid-based pathfinding. *Computational Intelligence and AI in Games*, 4(2): 144–148.