# Abstract: Block A* and Any-angle Path-Planning

**Peter Yap** and **Neil Burch** and **Robert C. Holte** and **Jonathan Schaeffer**
Computing Science Department
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
blueberrypete@gmail.com, {nburch, rholte, jonathan}@ualberta.ca

**Abstract**

We present three new ideas for grid-based path-planning algorithms that improve the search speed and quality of the paths found. First, we introduce a new type of database, the Local Distance Database (LDDB), that contains distances between boundary points of a local neighborhood. Second, an LDDB-based algorithm is introduced, called Block A*, that calculates the optimal path between start and goal locations given the local distances stored in the LDDB. Third, our experimental results for any-angle path planning in a wide variety of test domains, including real game maps, show that Block A* is faster than both A* and the previously best grid-based any-angle search algorithm, Theta*.

## 1 Introduction

In a grid-based pathfinding problem, the search space is usually stored as a two-dimensional array, where each cell can be accessed by its Cartesian coordinates. In the simplest (and most common) case, each cell has a binary value: obstructed (value 1) or unobstructed (value 0). Traditional grid searches using A* results in un-human-like paths that are confined to $45^\circ$ heading changes. The solution is to search for paths using algorithms that find smooth paths that can be "any-angle". We use this grid based path planning problem as the setting to introduce our new research ideas.

This paper summarize the results of (Yap et al. 2011) which makes the following contributions:

1. We generalize A* so that instead of expanding a node per iteration, we expand a block ($m \times n$ region of nodes) per iteration. We call this new algorithm Block A*; A* is a special case of Block A* that uses a $1 \times 1$ block.
2. Block A* uses a new type of database, the Local Distance Database (LDDB), that stores the all-pairs-shortest-path information between every pair of boundary block nodes, for every type of block pattern.
3. Block A* is data-driven in the sense that it uses the LDDB to compute $g$-values. By changing the LDDB, different types of searches can be performed (e.g., Manhattan, octile, or any-angle paths). Once computed, one LDDB can solve grid-maps of every size and configuration.

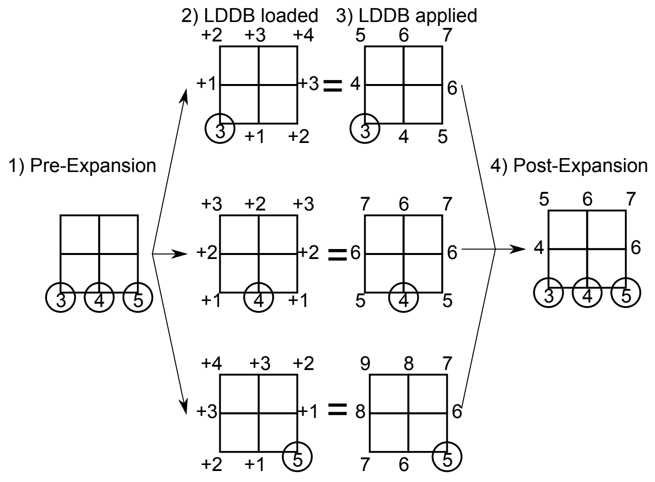

Figure 1: Expanding a block using the LDDB.

A* and Block A* mainly differ in how each expand a node (A*) or a block (Block A*). Consider Figure 1, where we expand a $2 \times 2$ block of unobstructed tiles. For simplicity, assume 4-way vertex moves with a zero heuristic. *1) Pre-Expansion:* This block's parent is from the south. The **ingress** vertices (circled) are the vertices from the parent block that have a finite $g$-value. The **egress** vertices are all the vertices on the block boundary. *2) LDDB loaded:* The LDDB entry for this block pattern is retrieved, $LDDB(y, x)$ returns the length of the shortest path between any ingress $y$ (circled) and egress $x$ for a given block obstacle pattern. Since we have three ingress vertices, we use the LDDB for each. The top block is the LDDB entry for the leftmost ingress ($g$=3). The middle block is the LDDB entry for the middle ingress ($g$=4). The bottom block is the LDDB entry for the rightmost ingress ($g$=5). *3) LDDB applied:* We add $y.g$ and $LDDB(y, x)$ to find the $g$-value of the shortest path from $start$ to $x$ via $y$. *4) Post-Expansion:* For each egress $x$, its new $g$-value is the minimum of its old $g$-value and the smallest $g$-value from all paths via these ingress (circled). $x'.g \; = \; min_{y \in Y}(x.g, y.g + LDDB(y, x))$. *e.g.,* shortest path to the top-left vertex is $min(5, 7, 9) = 5$. Finally, we place the four block neighbors of the expanded block into the open list using the $min_{updated\ s'}(s'.g + s'.h)$ as its heap value, where $s'$ are the boundary vertices shared by the

| Data Set | Algorithm | Distance | Time (s) |
|---|---|---|---|
| Random 0% | A* | 274.7 | 0.00481 |
| | Theta* | 260.8 | 0.00650 |
| | **Block A*** | 261.8 | **0.00103** |
| Random 10% | A* | 275.3 | 0.00489 |
| | Theta* | 261.6 | 0.00417 |
| | **Block A*** | 262.5 | **0.00140** |
| Random 20 % | A* | 276.4 | 0.00499 |
| | Theta* | 263.3 | 0.00494 |
| | **Block A*** | 264.3 | **0.00185** |
| Random 30% | A* | 277.5 | 0.00518 |
| | Theta* | 265.4 | 0.00632 |
| | **Block A*** | 266.6 | **0.00240** |
| Random 40% | A* | 282.7 | 0.00584 |
| | Theta* | 271.5 | 0.00904 |
| | **Block A*** | 273.0 | **0.00315** |
| Random 50% | A* | 296.9 | 0.00825 |
| | Theta* | 286.2 | 0.01484 |
| | **Block A*** | 287.8 | **0.00468** |
| Starcraft (random) | A* | 300.2 | 0.01268 |
| | Theta* | 285.7 | 0.11304 |
| | **Block A*** | 286.8 | **0.00506** |
| Baldur's Gate 2 (scenarios) | A* | 248.7 | 0.00334 |
| | Theta* | 237.2 | 0.01796 |
| | **Block A*** | 238.0 | **0.00147** |
| Dragon Age (scenarios) | A* | 409.0 | 0.00478 |
| | Theta* | 392.3 | 0.02697 |
| | **Block A*** | 393.9 | **0.00226** |

Table 1: Comparing algorithm performance

neighboring block and the expanded block.

In the domain of any-angle search, paths are not confined to 45° heading changes and are "any-angle". Both Theta* (Daniel et al. 2010) and Block A* do not guarantee the ground-truth optimal path, however their solutions are always better than or equal to A*'s path. Theta* is exactly the same as A* except it does an extra line of sight (LOS) check per node expansion. Thus it is always slower than A* per expansion. Block A* avoids this expensive LOS check by relying on its LDDB.

## 2 Experimental Results

We start experimenting on a $500 \times 500$ grid filled with randomly placed obstacles, with the probability of a cell being an obstacle ranging from 0% to 50%. In all our experiments, Block A* used the same $5 \times 5$ vertex block LDDB (1.2s to compute; size 60MB). All algorithms used the Euclidean distance heuristic.

All entries in the top part of Table 1 are averaged over 500 randomly generated obstacle maps, each with 100 path-planning problems based on random start and goal vertices, for a total of 50,000 data points. The bottom part of Table 1 is based on game maps. All results were obtained on a Core 2 2.8 GHz computer with 8 GB of memory. The time results are statistically significant at 99% confidence.

For random maps, Block A* gives the best trade-off between path quality and runtime. In path quality, Theta* gives the shortest paths, but is at best only 0.5% better than Block A*. In runtime, Block A* is always the fastest algorithm, it

| Map (size) | Algorithm | Distance | Expanded | Time (s) |
|---|---|---|---|---|
| 1000x1000 | A* | 518.3 | 478457 | 0.38 |
| | Theta* | 491.3 | 478096 | 22.57 |
| | **Block A*** | 493.1 | 30032 | **0.04** |

Table 2: Using a zero heuristic in a large unobstructed area.

is always faster than A*, at least twice as fast. In contrast, Theta*'s runtime significantly degrades as the number of obstacles increases, particular after the 30% obstacle range which reduces the effectiveness of the Euclidean heuristic. As Theta* is slower per expansion than A*, its performance diminishes when a poor heuristic causes it to search more nodes. In particular, consider the case of a large random map while searching with a zero heuristic. Table 2 compares the algorithms on a $1000 \times 1000$ grid using a zero heuristic with 100 random start/goal pairs. Block A* is fastest, over 560 times faster than Theta* while giving path solutions of comparable quality. Block A* has the benefit of being less sensitive to a bad heuristic compared to A* and Theta*.

Finally, Block A* has also been applied to traditional tile (4-way) and octile (8-way) pathfinding on game maps. By changing the LDDB pre-computation, different constraints are encoded into the search. In these experiments (not shown), Block A* was consistently 2-3-fold faster than A*.

## 3 Conclusions

Block A* is a generalization of A* such that it expands a block of nodes, instead of one node, per iteration. Its speed gain derives from its smaller search tree using blocks and pre-computed information stored in its LDDB.

From our experiments using real maps used in commercial games, Block A* is consistently faster than both A* and Theta* in both clogged and open areas. The real-time nature of games demand a fast path planning algorithm; in the industry A* is considered too slow. Theta* is always slower than A* per expansion, and much slower than A* when the heuristic is bad, when the map is open or both. In contrast, Block A* is always faster than A*. For all these criteria that make path planning difficult in games—poor heuristics, and real-time constraints—Block A* is always faster than both Theta* and A*. As well, Block A* will always find shorter and more realistic paths compared to A*; paths comparable to the slower Theta*.

## 4 Acknowledgments

We thank the reviewers for their comments; and Nathan Sturtevant and Dave Churchill for the game maps. This research was funded by NSERC and iCORE.

## References

Daniel, K.; Nash, A.; Koenig, S.; and Felner, A. 2010. Theta*: Any-angle path planning on grids. *JAIR* 39:533–579.

Yap, P.; Burch, N.; Holte, R.; and Schaeffer, J. 2011. Block A*: Database-driven search with applications in any-angle path-planning. In *AAAI*.