Block A*: Database-Driven Search with Applications in Any-angle Path-Planning

Peter Yap, Neil Burch, Rob Holte and Jonathan Schaeffer

Computer Science Department, University of Alberta, Canada

Presenter: Tugcem Oral

September 23, 2011

Outline

Introduction

- Paper published in AAAI' 11
- Fast path planning is challenging due to:
  - Behaviour of dynamic environments.
  - Domain may require paths to be computed for multiple agents.
- Grids are standart for highly dynamic multi-agent domains
- Moving from cell to another can be done in several ways
  - Navigate 4 closest neighbours - 90º turns (tile)
  - Navigate 8 closest neighbours - 45º turns (octile)
  - Navigate on any-angle grid cell. (Field D* and Theta*)

Introduction (cont'd)

- Three new ideas for grid-based path-planning
  - Local Distance Database (LDDB)
  - Block A* Algorithm
  - Block A* is faster than A* and Theta*, the *previous* best grid based any-angle search algorithm.

Overview

- Stores the exact distance between the boundary points of a local region.
- The search space is grouped into regions of *m x n* contiguous cells.
- During a search, LDDB is queried to find *g-values.*

Efficiency in LDDB

- Search space is stored in 2D array
  - For a *b x b* block of cell, there are 2^{bxb} possible grid obstruction
  - This constraint is handled in LDDB, a *4x4* block LDDB is already very effective.
- Symmetry can be used to reduce the number of entries of LDDB.
- For most domains, there exists unreachable cells and these can be eliminated.
- An optimal path cost will be stored for every boundary cell of the block to every other boundary cell on all four sides of the block.

Overview

- Block A* is A* adapted to manipulate a block of cells instead of a single cell at a time.
- Each entry on its OPEN list is a block that has been reached but not yet expanded, or which needs to be re-expanded because new or cheaper paths to it have been found.
- The priority of a block on the OPEN list is called its heap value.
- Like A*, the basic cycle in Block A* is to remove the OPEN entry with the lowest heap value and expand it.
    - The LDDB is used during expansion to compute g-values.

Expansion of a Block

- *Only horizontal and vertical moves.*
- Ingress cells are boundary cells of actual block (*set of Y*)
- First Step: identify *valid egress* cells.
- Calculate g-values between ingress and egress cells;
  - $x_g = min_{y \in Y}(y_g + LDDB(y, x), x_g)$
- Compute heap value for each neighbouring block.

Expansion Example

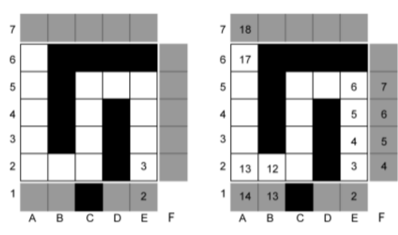


Figure: Expanding a Block - before and after

Expansion of a Block (cont'd)

Algorithm 1 Expand *curBlock*. Y is the set of *curBlock*'s ingress cells.

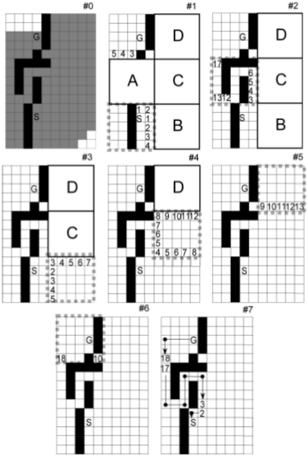
PROC: Expand($curBlock$, Y)
for side of $curBlock$ with neighbor $nextBlock$ **do**
    for valid egress node x on current side **do**
        x' = egress neighbor of x on current side
        $x.g = min_{y \in Y}\ (y.g + \mathrm{LDDB}(y, x), x.g)$
        $x'.g = min(x'.g, x.g + \mathrm{cost}(x, x'))$
    end for
    $newheapvalue = min_{\mathrm{updated}\ x'}\ (x'.g + x'.h)$
    if $newheapvalue < nextBlock$.heapvalue **then**
        $nextBlock$.heapvalue = $newheapvalue$
        if $nextBlock$ not in OPEN **then**
            insert $nextBlock$ into OPEN
        else
            UpdateOPEN($nextBlock$)
        end if
    end if
end for

The Block A* Algorithm

Algorithm 2 Block A*

PROC: Block A* (LDDB, $start$, $goal$)
$startBlock = init(start)$
$goalBlock = init(goal)$
$length = \infty$
insert $startBlock$ into $OPEN$
while ($OPEN \neq empty$) **and** (($OPEN$.top).heapvalue $< length$)) **do**
    $curBlock = OPEN$.pop
    Y = set of all $curBlock$'s ingress nodes
    if $curBlock == goalBlock$ **then**
        $length = min_{y \in Y}(\, y.g + dist(y, goal), length)$
    end if
    Expand($curBlock, Y$)
end while
if $length \neq \infty$ **then**
    Reconstruct solution path
else
    return Failure
end if

Block A* Example



Any-Angle Search

- Search operate on vertices.
  - But obstacles are still cell-based.
- To use Block A* search on vertices, only construct a LDDB that takes exterior vertices of a block as input, rather than using the exterior cells.
  - Algorithm does not change!
- For vertex blocks, the exterior vertices of a block are shared with another block and the corner vertices are shared with 3 other blocks.

Example of Any-Angle Search

- The shortest path from B1 to E6 using cells is 8.
- With an any-angle search;
    - Block A* finds the optimal path B1 – D4 – E6 (dashed line) with a cost of 5.84
    - Theta* will find the suboptimal path B1 – E5 – E6 after some computation (open arrows) with a cost 6.
    - A* will find a zig-zag-like path B1 – C2 – C3 – D4 – D5 – E6 (filled circles) with a cost 6.24

Any-Angle Search Representation

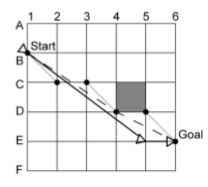


Figure: Any-angle search results on a vertex block

Experimental Results

- 500×500 grid filled with randomly placed obstacles
  - Obstacle probability ranging from 0% to 50%.
- In all experiments, Block A* used the same 5×5 vertex block
- Starcraft (RTS), Baldur's Gate 2 and Dragon's age (FRP): Origins maps are used on experimental results.

Experimental Results Table

Data Set	Algorithm	Distance	Expanded	Time (s)
Random 0%	A*	274.7	14957	0.00481
	Theta*	260.8	918	0.00650
	Block A*	261.8	638	**0.00103**
Random 10%	A*	275.3	15039	0.00489
	Theta*	261.6	4439	0.00417
	Block A*	262.5	845	**0.00140**
Random 20 %	A*	276.4	15351	0.00499
	Theta*	263.3	6229	0.00494
	Block A*	264.3	1159	**0.00185**
Random 30%	A*	277.5	15889	0.00518
	Theta*	265.4	8536	0.00632
	Block A*	266.6	1617	**0.00240**
Random 40%	A*	282.7	18025	0.00584
	Theta*	271.5	12603	0.00904
	Block A*	273.0	2407	**0.00315**
Random 50%	A*	296.9	26146	0.00825
	Theta*	286.2	22721	0.01484
	Block A*	287.8	4476	**0.00468**
Starcraft (random)	A*	300.2	26456	0.01268
	Theta*	285.7	23729	0.11304
	Block A*	286.8	2890	**0.00506**
BG2 (scenarios)	A*	248.7	10796	0.00334
	Theta*	237.2	7043	0.01796
	Block A*	238.0	1034	**0.00147**
DA:0 (scenarios)	A*	409.0	15465	0.00478
	Theta*	392.3	14478	0.02697
	Block A*	393.9	1709	**0.00226**

Why Block A* is Better?

- Block A* benefits from the pre-computed results in LDDB to avoid work.
- The darker the diagram, the more computations needed.

Overheads on Algorithms

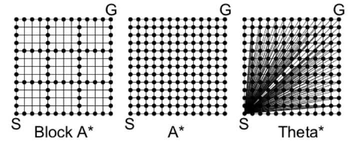


Figure: Overheads

Conclusion

- Block A* performs well with both good and bad heuristics, and is always faster than both A* and Theta*.
- Block A* will always find shorter and more realistic paths compared to A*; paths comparable to the slower Theta*
- Theta* may be good for low obstructed areas, but flounders in open areas.
- A map having open and clogged areas (or can dynamically change) makes deciding which algorithm to use.

Q & A

Any Questions?

- Daniel, K.; Nash, A.; Koenig, S.; and Felner, A. 2010. Theta*: Any-angle path planning on grids. JAIR 39:533– 579.
- Ferguson, D., and Stentz, A. 2006. Using interpolation to improve path planning: The field D* algorithm. Journal of Field Robotics 23(2):79–101.
- Lozano-P´ rez, T., and Wesley, M. 1979. An algorithm for e planning collision-free paths among polyhedral obstacles. Communications of the ACM 22(10):560–570.
- Millington, I., and Funge, J. 2009. Path smoothing. In Artificial Intelligence in Games: 2nd Edition, 251–256.
- Nash, A.; Koenig, S.; and Tovey, C. 2010. Lazy theta*: Any-angle path planning and path length analysis in 3d. In AAAI.