Breaking Path Symmetries on 4-Connected Grid Maps.

Conference Paper · January 2010

Source: DBLP

CITATIONS
16

READS
136

2 authors:

Daniel Harabor
National ICT Australia Ltd
17 PUBLICATIONS **164** CITATIONS
SEE PROFILE

Adi Botea
IBM
69 PUBLICATIONS **1,004** CITATIONS
SEE PROFILE

Breaking Path Symmetries on 4-connected Grid Maps

Daniel Harabor and **Adi Botea**
NICTA and The Australian National University
Email: firstname.lastname@nicta.com.au

Abstract

Pathfinding systems that operate on regular grids are common in the AI literature and often used in real-time video games. Typical speed-up enhancements include reducing the size of the search space using abstraction, and building more informed heuristics. Though effective each of these strategies has shortcomings. For example, pathfinding with abstraction usually involves trading away optimality for speed. Meanwhile, improving on the accuracy of the well known Manhattan heuristic usually requires significant extra memory.

We present a different kind of speedup technique based on the idea of identifying and eliminating symmetric path segments in 4-connected grid maps (which allow straight but not diagonal movement). Our method identifies rectangular rooms with no obstacles and prunes all interior nodes, leaving only a boundary perimeter. This process eliminates many symmetric path segments and results in grid maps which are often much smaller and consequently much faster to search than the original. We evaluate our technique on a range of different grid maps including a well known set from the popular video game Baldur's Gate II. On our test data, A* can run up to 3.5 times faster on average. We achieve this without using any significant extra memory or sacrificing solution optimality.

Introduction

In the context of single-agent pathfinding A* (Hart, Nilsson, and Raphael 1968) is regarded as the gold standard search algorithm. It is complete, optimal and optimally efficient which makes it very attractive to researchers in the area. Many studies exist which have attempted to improve on the performance of A*. The majority focus in one of two directions: reducing the search space through hierarchical decomposition and identifying better heuristics to guide search. In the case of hierarchical decomposition, techniques such as HPA* (Botea, Müller, and Schaeffer 2004) and PRA* (Sturtevant and Buro 2005) seek to construct and explore a much reduced approximate state space. These methods are fast and require no significant extra-memory when compared to A*. However, they have the disadvantage that solutions are not guaranteed to be optimal. Meanwhile, in case of the improved heuristics, it has been frequently shown that obtaining better informed results than than the popular Manhattan heuristic usually incurs significant memory overhead (Sturtevant et al. 2009; Goldberg and Harrelson 2005; Cazenave 2006; Björnsson and Halldórsson 2006). Furthermore it is well known that even heuristics which differ from perfect information by at most a (small) additive constant, can still exhibit poor performance on a range of problems such as AI planning and graph search (Helmert and Röger 2008; Pohl 1977).

In this paper we explore a new speedup technique that aims to reduce the size of the search space while preserving optimality. Our work focuses on eliminating symmetric path segments from 4-connected grid maps, which allow straight but not diagonal movement. Although less popular than the 8-connected variant, this domain appears regularly in the literature (Yap 2002; Wang and Botea 2008; Pochter, Zohar, and Rosenschein 2009) and is often found in the pathfinding systems of modern video games. Some recent examples include Square Enix's *Heroes of Mana* (released in 2007 for the Nintendo DS), Astraware's *My Little Tank* (2008, iPhone) and Atari's *Dragon Ball Z: Legacy of Goku* (2002, Gameboy Advance).

Consider as a motivating example the simple map in Figure 1. Such topographies, with rooms and corridors, appear often in video games (e.g., dungeon areas in roleplaying games can be described in such terms, albeit on a larger scale). Running A* on a standard grid can be surprisingly inefficient in such instances. In Figure 1, many tiles have *f-values* smaller than the goal's, and A* must expand them. Many of the explored paths are symmetric in the sense that they can be obtained from each other by re-ordering the moves.

To solve the problem more efficiently we will automatically discover obstacle-free rectangular rooms and observe that while there are many ways to optimally traverse across a room there is always a solution involving only nodes from the perimeter. This observation forms the basis for our symmetry elimination technique: we prune all nodes from the interior but not the perimeter of an empty room and, to preserve optimality, replace them with a series of *macro edges* that allow moving directly from one side of an empty room to the other. In the process we eliminate almost all equivalent path segments when crossing a room, significantly re-

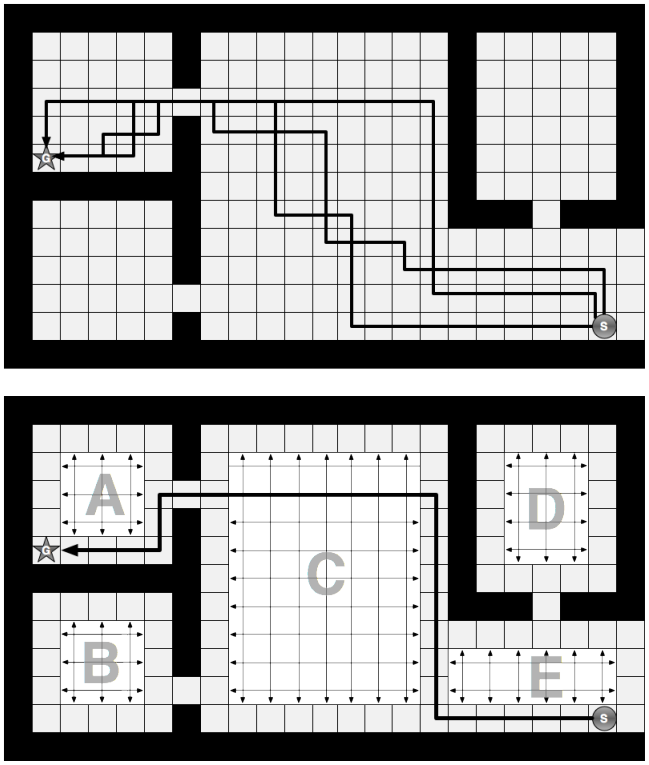

Figure 1: (Top) A highly symmetric pathfinding instance. Many solutions exist; we highlight three. (Bottom) The same map with our symmetry elimination method applied. We decompose the map into a set of obstacle-free rooms from which we prune all nodes except those on the perimeter. We replace them with a small set of macro edges that connect the perimeter nodes directly.

ducing the number of locations that must be explored. Cases when the start (or the goal) location belongs to the interior of an empty rectangle are handled by temporarily re-inserting these nodes back into the map.

Our method is easy to understand and implement. It produces grid maps that are often much smaller than the original grid map (and never larger), have the same branching factor and preserve the same completeness and optimality guaranteeing characteristics. Further, since our method is focused on eliminating symmetries in the search space, it is orthogonal to, and can be used in conjunction with, existing search techniques including hierarchical and low level pathfinding systems as well as memory-based heuristics. We demonstrate its effectiveness by undertaking an empirical analysis on a wide range of maps, including a well known set from the popular roleplaying game *Baldur's Gate II*. On our test data the average performance of A* is increased by a factor of up to 3.5, depending on the topography of the map being used.

Related Work

One recent result on the problem of eliminating unnecessary path fragments on grid maps is due to Pochter, Zohar, and Rosenschein (2009). They introduce *swamps*, which are areas that don't have to be searched because crossing them would not improve the length of paths. The identification of swamps is quite different to our empty room decomposition. Additionally, swamps are shown to be most effective in areas featuring a large number of obstacles and less effective on maps featuring wide open areas. By comparison, our algorithm is most effective when large empty rooms can be identified and less effective when this is not the case. Thus the two methods are in some sense complementary.

Our work also bears some similarity to new heuristic methods aimed at improving the performance of standard A* on grid maps (Björnsson and Halldórsson 2006). In that work, like in ours, grid maps are decomposed into obstacle-free zones connected by entrances and exits. A preliminary online search in the decomposed graph identifies zones that do not appear on any path between the start and goal node, thus yielding the *dead-end heuristic*. It can be seen as a technique for detecting areas that don't have to be searched in the instance at hand and, as before, is complementary to our work.

MSA* (Bolanca 2009) is a new optimality preserving search algorithm which attempts to speed up search on 8-connected grid maps by exploiting path equivalence in empty rectangular rooms. Rather than pruning nodes from the interior of an empty room however MSA* attempts to speed up search by generating macro edges on the fly. An improvement over conventional A* is reported but the algorithm is also shown to expand a large number of nodes from the interior of empty rooms, which hampers its performance. It is also worth noting that MSA* uses a different empty room decomposition method from the one described in our work.

Fringe Search (Björnsson et al. 2005) is a general purpose iterative deepening technique which also aims to improve on the performance of A*. This work is quite different from others we have discussed in that it does not rely on any specific decomposition technique nor on the development of any new heuristics to guide the search. It is provably optimal if maximum search depth is sufficiently large and it has been shown to run between 25-40% faster than A*. As our work is an offline graph-pruning technique, it could be combined with any search algorithm, including Fringe Search.

Another effective method for solving path planning problems is to reformulate the original problem into an equivalent one in a much smaller abstract search space. Algorithms in this category are usually fast, memory-efficient and sub-optimal. The HPA* algorithm (Botea, Müller, and Schaeffer 2004) uses a map decomposition approach, dividing a grid map into a series of fixed-size clusters connected by entrances. As with Fringe Search, HPA* could be combined with our work on 4-connected grids. For example, first apply our pruning strategy and then apply HPA* to the resulting grid map.

Offline Symmetry Elimination

Pathfinding in modern video games often involves exploring highly regular environments such as cities, sewers or dungeons (e.g Figure 4). Though these locales tend to be topo-

graphically simple (usually being comprised of empty rooms connected by corridors) they can also be highly symmetric with many optimal length paths existing between arbitrary pairs of locations. Symmetry appears in many domains (e.g. constraint programming (Walsh 2007)) and, unless it is handled properly, almost always increases the size of the search space and forces search algorithms to waste time.

We propose the following offline strategy for identifying and eliminating symmetric paths in 4-connected grid maps:

1. Decompose the grid map into a set of empty rooms, where each empty room is rectangular in shape and free of any obstacles. The size of the rooms can vary across a map, depending on the placement of the obstacles.
2. Prune all nodes from the interior but not the perimeter of each empty room.
3. Add a series of *macro edges* that connect each node on the perimeter of an empty room with a node on the directly opposite side of the room [1]. The cost of each edge is equal to the Manhattan distance between its two endpoints.

Trivial rooms which contain no interior nodes (for example rooms with a width w or height $h \leq 2$) are left unmodified by steps 2 and 3. Figure 1 shows an example of this process. For each non-trivial room we prune $(w-2) \times (h-2)$ interior nodes and, in the process, eliminate a large number of symmetric paths between nodes on the perimeter. We claim that this approach preserves optimality when traversing across any arbitrary room.

Lemma 1. *Let R be an arbitrary rectangular room that is free of obstacles and m, n be two locations on its perimeter. Then m and n can be connected optimally through a path that mentions only nodes on the perimeter of R and possibly involves a macro edge.*

Proof. There are two distinct cases to consider. Case 1 is when m and n are placed on the same side of the perimeter, or on two orthogonal sides. To obtain an optimal path we can simply travel along the perimeter from m to n. Case 2 is when m and n are placed on opposite sides of the perimeter. To obtain an optimal path we can simply follow the macro edge at m and navigate directly to a node m' located on the same side of the perimeter as n. Then, go from m' to n along the perimeter. The resultant path is optimal as its length is equal to the Manhattan distance between m and n. □

A direct corollary to Lemma 1 is that we can prune from consideration all nodes from the interior of R and limit ourselves to only searching nodes appearing along its perimeter. The only remaining consideration is how to deal with interior nodes that happen to be the start or goal location for the search at hand. We address this case in the following section.

Online Insertion

Often an interior node pruned as a result of offline symmetry elimination is required as a start or goal location for an agent. We handle such cases by inserting back into the graph interior start and goal nodes for the duration of the search. We use the following procedure (highlighted in Figure 2):

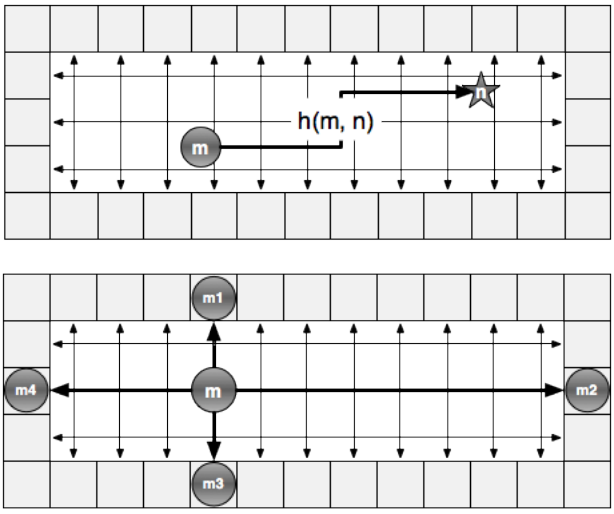

Figure 2: (Top) When m and n are in the empty room no insertion is necessary. (Bottom) m is a previously pruned interior node. We insert m into the graph and connect it to neighbours on each side of the empty room.

1. If the start and goal are in the same room no insertion is required. Since it is guaranteed that there are no obstacles between the two locations, an optimal path is trivially available. This case will be ignored in the rest of our discussion.
2. If the start and goal are not in the same room, connect each of them to the closest neighbours on each side of the perimeter of the empty room.

We claim that this procedure retains optimality when searching from the start (or goal) location to all nodes on the perimeter of its room.

Lemma 2. *Let R be an empty rectangular room. For any nodes m, n, with m a re-inserted interior node and n a node on the perimeter, it is always possible to find an optimal length path which mentions no interior nodes except for m.*

Proof. We insert m into the graph and connect it to m'_1, m'_2, m'_3, m'_4, the closest neighbours on each side of the perimeter. The weight of each edge incident with m is equal to the Manhattan distance between m and each m'_i. To find an optimal path to n we travel from m to the node m'_i which is on the same side as n on the perimeter. From there we travel along the perimeter of R until we reach n. □

Once the search has finished we remove the start and goal from the graph. The time required in each case (insertion and deletion) is constant.

Optimality

We claim that the symmetry elimination procedure outlined earlier is sufficient to guarantee that A*, running on our pruned grid maps, will always return an optimal solution if one exists.

[1] Alternatively, macro edges could be generated on-the-fly during search. This obviates the need to store them explicitly.

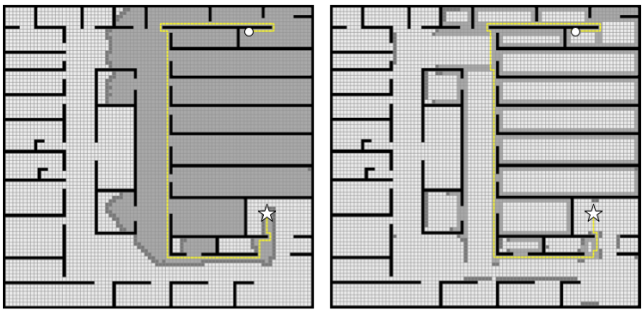

Figure 3: (Left) A* solving a problem on an unmodified (86×88) grid map. Expanded nodes are marked dark grey. (Right) A* solving the same problem using our modified grid map. The algorithm only considers nodes along the perimeter of the identified rooms.

Theorem 3. *For every optimal length path $\pi^*(s, g)$ in a 4-connected grid map there exists an equivalent length path in the pruned version of the grid map.*

Proof. Follows from Lemma 1 and Lemma 2. For every optimal length segment of $\pi^*(s, g)$ which traverses through an empty room, from a perimeter node m to a perimeter node n, there is an equivalent segment which mentions only nodes on the perimeter of that room (and possibly one macro-step). □

A direct corollary of Theorem 3 is that optimal solutions pruned by our symmetry reduction can be easily reconstructed; for example to avoid unnatural looking paths where agents seem to hug walls. Consider a path fragment between m and n, two nodes on the perimeter of an empty room. Assume, without any generality loss, that the path fragment contains r moves to the right and u moves upwards. All optimal path fragments between m and n can be obtained by interleaving r moves to the right and u moves upwards in any order (e.g right-right-up, right-up-right, up-right-right)

In Figure 3 we highlight the effectiveness of our symmetry breaking technique using a map that has characteristics typical of what one might expect in a modern role-playing game[2]; there are many rooms and corridors and many entrances connecting them. A* running on the original grid map expands almost half the nodes in the state space of the shown example. We then apply our technique to eliminate symmetries and re-run A*. This time A* expands less than 15% of all nodes (more than a three-fold improvement) and returns an optimal solution 3 times faster.

Identifying Empty Rooms

In this section we give a simple but effective flood-fill-based algorithm for decomposing a grid map into empty rectangular rooms. We will try to build large rooms before small ones and prefer rooms which contain as many interior nodes as possible:

1. For each traversable tile t, build a maximal size empty rectangle which has t as its upper left corner. Each such rectangle should contain only traversable tiles which have not already been assigned to a room.
2. Using a Max-Heap, sort the list of traversable tiles using the number of interior nodes in the rectangle of each t as its priority.
3. Take from the heap the tile t with highest priority which has not already been assigned to a room.
4. Verify the priority of t by building another maximal size empty rectangle (as per Step 1) which has t as its upper left corner and contains no obstacles or tiles already assigned to another room.
5. If the number of interior nodes in the new rectangle is equal to the priority of t we say that the rectangle forms a room and add it to our decomposition. Otherwise, we update the priority of t with the number of interior nodes contained in the new rectangle.
6. Repeat Steps 3 to 5 until the heap is empty and all nodes have been assigned to a room in the decomposition.

The construction of empty rooms is similar to the computation of *clearance values* in Harabor and Botea (2008). In that work the objective is to calculate the amount of traversable space at any given location on the map. This is achieved by constructing maximum sized squares that originate at each traversable tile on the map. Our room identification procedure can be seen as a variation of this method in which we extend each such square into a maximal size rectangle.

As we will see the performance of A* on our modified grid map is closely related to the total number of nodes we are able to prune. Thus, identifying large rooms is critical. Although our decomposition technique is not optimal for this purpose it is simple to understand and implement and produces good results in practice.

Experimental Setup

To evaluate the effectiveness of our symmetry elimination technique we performed a comparative analysis using A* on a number of benchmarks taken from the freely available pathfinding library Hierarchical Open Graph (HOG)[3]:

- **Adaptive Depth** is a set of 12 maps of size 100×100 in which approximately $\frac{1}{3}$ of each map is divided into adjacent rectangular rooms of varying size and the rest of the map is a large open area interspersed with large randomly placed obstacles.
- **Baldur's Gate** is a set of 120 maps taken from BioWare's popular roleplaying game *Baldur's Gate II: Shadows of Amn*. Often appearing as a standard benchmark in the literature (Botea, Müller, and Schaeffer 2004; Björnsson et al. 2005; Björnsson and Halldórsson 2006; Sturtevant and Buro 2005; Harabor and Botea 2008) these maps range in size from 50×50 to 320×320 and have a distinctive 45-degree orientation. Figure 4 shows a typical example.

[2] In fact, many video game maps tend to be somewhat bigger than our example but for demonstration purposes it is sufficient.

[3] `http://www.googlecode.com/p/hog2`

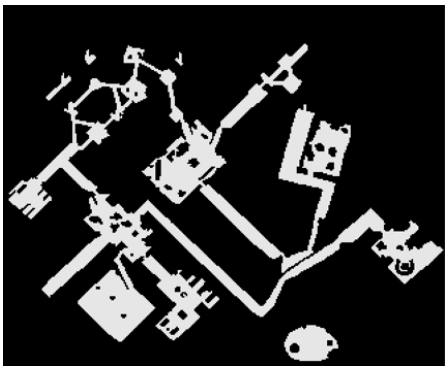

Figure 4: A map from BioWare's *Baldur's Gate II*

- **Rooms** is a set of 300 maps of size 256×256 which are divided into 32×32 rectangular areas that are connected by randomly placed entrances.

For each map we removed all diagonal edges and randomly generated 100 valid problem instances. We then ran A* twice: once on the original maps and again on our modified maps making for a total of 86400 (432×100×2) distinct experiments. Our test machine had a 2.93GHz Intel Core 2 Duo processor, 4GB RAM and ran OSX 10.6.2. We use the A* implementation provided in HOG.

Results

Our main results are given in Table 1 and Figure 5. We will briefly introduce both and then discuss them in the context of each of our three benchmarks.

Table 1 measures the percentage of nodes pruned from each set of maps. We give figures for the minimum, maximum and average number of nodes pruned. We also give the standard deviation as an indicator for the level of variability associated with each result. Meanwhile, Figure 5 shows the average speedup experienced by A* when running on our modified grid maps as compared to the original. We measure speedup in terms of node expansions and search times. For example, a search time speedup of 2.0 is twice as fast and a node expansion speedup of 2.0 indicates 50% fewer nodes were expanded.

Table 1: Graph Size (% of Nodes Pruned)

Benchmark	Mean	Min	Max	Std Dev.
BG	42.33	19.82	78.36	11.80
AD	63.24	57.98	66.69	2.10
Rooms	49.77	49.46	49.96	0.19

Adaptive Depth: The topography of the maps in this benchmark were favourable for our symmetry breaking technique. Our decomposition algorithm was able to identify many large open areas and pruned between 58% to 67% of all nodes. Its average performance was just over 63%. We also observed up to a factor of 3.5 reduction in the average number of nodes expanded by A* and a similar maximum search time speedup. It is interesting to note that for short instances, for example those with path lengths < 15, we observed search times that were only 1.7 times faster on average. By comparison, instances with longer path lengths were solved 2.6 to 3.5 times faster. This is because in many cases, though not in general, longer paths tended to traverse through more empty rooms where there exists more opportunities to take advantage of the pruning enhancement.

Baldur's Gate: The maps in this benchmark are a mixture of large and small areas which sometimes contain obstacles and may be connected by long narrow corridors. They also have a distinct 45-degree orientation which makes it difficult to decompose traversable areas into rectangular rooms. For example, though our decomposition algorithm can prune as many as 78% of all traversable nodes on some maps, its average performance is only 42%. There is also a reasonably high level of variability associated with this result: we measured a standard deviation of almost 12%. Nevertheless, we observed that average A* search times and average A* node expansions were both improved by a factor of between 1.8 to 2.3. We expect that these results could be further improved given a more effective decomposition algorithm.

Rooms: The maps in this benchmark were all very similar, comprising of 32×32 rectangular rooms connected by randomly placed entrances. Each room is of size 7×7 and contains 49 nodes. 24 of these (or just under 50%) are interior nodes which we expected would be pruned. Table 1 shows that this was indeed the case. When we ran A* on these grid maps we observed in most cases a factor of 2 reduction in both the average number of nodes expanded and average search time. Given rooms with proportionally larger dimensions we would expect to see a proportionally larger improvement in the performance of A*. We expect the same is also true as rooms become smaller where in the worst case there are no interior nodes to prune from any room.

Conclusion

We study the problem of symmetric path elimination in 4-connected grid maps. Though less popular than the 8-connected variant, 4-connected grid maps appear regularly in modern video games and academic literature.

We presented a novel offline method for breaking path symmetries which is simple to understand and requires no significant extra memory. Our method involves decomposing a map into empty rectangular rooms, pruning all nodes appearing in the interior and replacing them with a set of *macro edges* that facilitate optimal traversal from the perimeter of any room to the perimeter of any other. We also give an online node insertion technique that extends these guarantees to arbitrary pairs of locations appearing in the original unmodified map.

We evaluate the performance of our algorithm by running A* on a wide range of realistic game maps including one well known set from the game *Baldur's Gate II*. In many cases we are able to prune over 50% of all nodes on a given map and improve the average search time performance of A* by a factor of up to 3.5.

The performance of our method depends on the topography of individual maps: in the presence of large rooms or

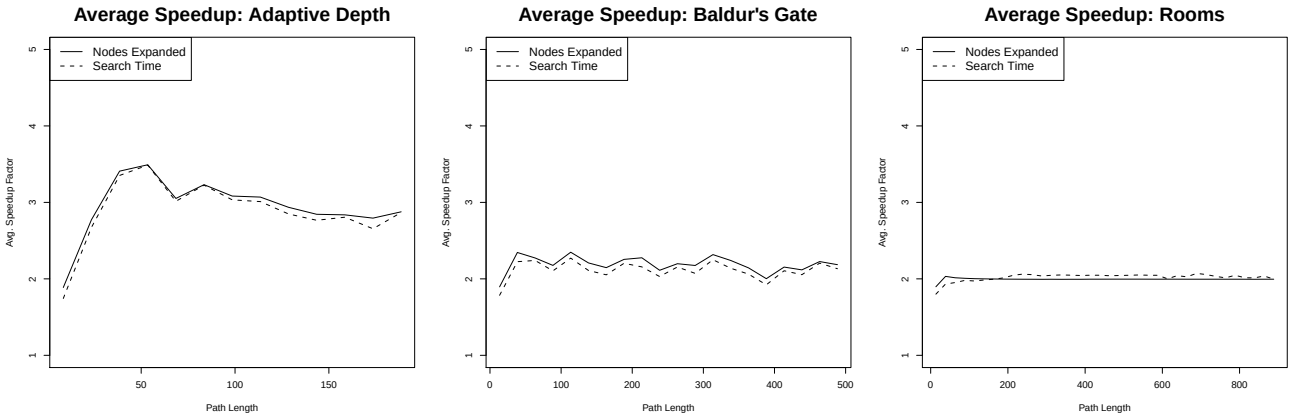

Figure 5: Average A* speedup on each of our three benchmarks. Results are given in terms of nodes expanded and search time.

wide open areas (both commonly seen in video games[4]) we can often compute optimal paths much faster than searching on the original map. On less favourable map topographies we achieve more modest improvements. However, since our method is orthogonal to existing search techniques, it could be integrated as part of a larger framework involving specialised heuristics or other speedup techniques; for example as described in (Botea, Müller, and Schaeffer 2004; Björnsson et al. 2005; Björnsson and Halldórsson 2006).

One direction for further work is to study breaking path symmetries in 8-connected grid maps. This domain also exhibits a high degree of path symmetry but the problem is more challenging because each tile has a higher branching factor. Another direction for future work is to investigate alternative decomposition algorithms which produce bigger rooms and improve the performance of the current method.

Acknowledgments

We thank Philip Kilby for the many engaging discussions and continued support during the development of this work. We also thank Alban Grastien for taking the time to review and comment on draft versions of this paper. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

Björnsson, Y., and Halldórsson, K. 2006. Improved heuristics for optimal path-finding on game maps. In *AIIDE*, 9–14.

Björnsson, Y.; Enzenberger, M.; Holte, R. C.; and Schaeffer, J. 2005. Fringe search: Beating A* at pathfinding on game maps. In *CIG'05*, 125–132.

Bolanca, M. 2009. Achieving fast and optimal pathfinding through the use of macro steps in obstacle free areas. Australian National University Honours Thesis.

Botea, A.; Müller, M.; and Schaeffer, J. 2004. Near optimal hierarchical path-finding. *J. Game Dev.* 1(1):7–28.

Cazenave, T. 2006. Optimizations of data structures, heuristics and algorithms for path-finding on maps. In *CIG*, 27–33.

Goldberg, A. V., and Harrelson, C. 2005. Computing the shortest path: A* search meets graph theory. In *SODA*, 156–165.

Harabor, D., and Botea, A. 2008. Hierarchical path planning for multi-size agents in heterogeneous environments. In *CIG'08*.

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):100–107.

Helmert, M., and Röger, G. 2008. How good is almost perfect? In *AAAI*, 944–949.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2009. Using swamps to improve optimal pathfinding. In *AAMAS*, 1163–1164.

Pohl, I. 1977. Practical and theoretical considerations in heuristic search algorithms. In Elcock, E. W., and Michie, D., eds., *Machine Intelligence 8*. Ellis Horwood Ltd and John Wiley & Sons.

Sturtevant, N. R., and Buro, M. 2005. Partial pathfinding using map abstraction and refinement. In *AAAI*, 1392–1397.

Sturtevant, N. R.; Felner, A.; Barrer, M.; Schaeffer, J.; and Burch, N. 2009. Memory-based heuristics for explicit state spaces. In *IJCAI*, 609–614.

Walsh, T. 2007. Breaking value symmetries. In *CP*, 880–888.

Wang, K.-H. C., and Botea, A. 2008. Fast and memory-efficient multi-agent pathfinding. In *ICAPS*, 380–387.

Yap, P. 2002. Grid-based pathfinding. In *LNCS*, volume 2338. 44–55.

[4]For example, Blizzard's popular multi-player game *World of Warcraft*