Expected Path Degradation when Searching over a Sparse Grid Hierarchy

Robert Kolchmeyer, Andrew Dobson, and Kostas E. Bekris
Department of Computer Science, Rutgers University
New Brunswick, New Jersey 08901

Abstract

The traditional focus of combinatorial search research is to speed up the search algorithm. An alternative, however, is to create a sparser representation of the search space. This relates to the idea of spanners from graph theory. These are subgraphs which retain paths between any two vertices of the original graph while guaranteeing a maximum stretch in path length. In practice, the path degradation of graph spanners is significantly smaller than the theoretical bound. Even so, expected path degradation of graph spanners is typically not studied. This work focuses on grid path-finding to propose an algorithm that constructs a grid spanner, where analysis for the obstacle-free case shows that significant performance gains can be achieved with a small decrease in expected path quality. This is an important first step towards studying the expected performance of spanners. Experiments on game maps show that expected path quality with obstacles is only sometimes marginally lower than that in the obstacle-free case and that a significant reduction in the size of the search space can be achieved.

Introduction

There is extensive research into improving the efficiency of A* (Hart, Nilsson, and Raphael 1968), such as suboptimal search (Thayer and Ruml 2011) or anytime methods that quickly return suboptimal paths and refine them over time (Likhachev, Gordon, and Thrun 2004). Lifelong planning (Koenig, Likhachev, and Furcy 2004) instead retains information to speed up subsequent searches, and is helpful in static or near-static environments.

An alternative approach follows a hierarchical representation of the search spaces to balance the trade-off between optimality and search speed (Uras and Koenig 2014; Pochter et al. 2010; Sturtevant and Buro 2005; Botea, Muller, and Schaeffer 2004; Storandt 2013). This is related to motion planning methods that provide approximate, sparse representations of the search space (Marble and Bekris 2011; Wang, Balkcom, and Chakrabarti 2013; Shaharabani et al. 2013; Dobson, Krontiris, and Bekris 2013; Marble and Bekris 2013; Dobson and Bekris 2014). Similar to this reasoning, this work studies the sparsification of an input grid during a preprocessing step and shows that by implicitly

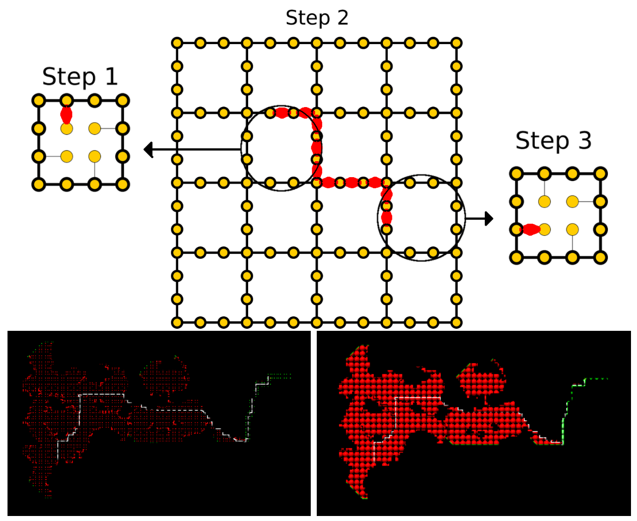

Figure 1: Paths on the spanner originate and terminate in cell structures, and travel via a highway system (step 2). The maps illustrate the proposed sparse construction.

constructing a *grid spanner* (Liestman and Shermer 1993), search time can be greatly improved. A motivating application is the construction of sparse representations for video game maps.

A *grid spanner* is defined to be a *graph spanner* (Peleg and Schäffer 1989) over a grid. A graph spanner of a graph $G = (V, E)$ is a subgraph $H = (V, E')$, such that shortest path lengths in H are bounded by the corresponding shortest path lengths in G, i.e., $(\forall_{u,v \in V}) d_H(u, v) \leq \alpha \cdot d_G(u, v) + \beta$ (Baswana et al. 2010). In general, such bounds provide a worst case multiplicative bound on path length, α, called the *stretch* of the spanner. A path's degredation will refer to the ratio of path length $d_H(u, v)$ over the original path length $d_G(u, v)$. While a graph spanner can only guarantee that no path has degradation larger than α, in practice, most paths are significantly better than the worst case. In fact, most paths are often very close to optimal. Thus, an expected case analysis of graph spanners provides a significantly better estimate of path quality.

Expected case analysis, however, remains largely unstudied for graph spanners. Previous work in grid spanners focused on the average degree of nodes, rather than the average stretch (Liestman and Shermer 1993). Work on the average stretch for spanning trees reasoned in terms of edge hops rather than edge weights (Elkin et al. 2008). As a step towards general expected case analysis for graphs, this work proposes a grid spanner construction and provides an expected-case analysis for an obstacle-free scenario. The proposed construction partitions the grid into cells, which are then sparsified, creating a hierarchical structure. Paths on this spanner traverse a sparser grid structure as in Figure 1.

To solve problems with obstacles, this paper describes a methodology that extends the grid spanner idea. Experimentally evaluating the cost of searching over the proposed grid spanner construction yields performance benefits with little path degradation, and correlates well with drawn approximations for the expected degradation. This results in a significant decrease in the size of the search space, while producing high-quality paths.

Obstacle-Free Case

This work considers a specific grid spanner construction, which will be generalized later. Using this construction, it will be possible to show how sparse the resulting structure is, and then proceed to analyze the expected path degradation for paths through the grid spanner.

Grid spanner construction: The construction partitions the grid into $k \times k$ cells, where k is an even natural number, sparsifies each cell, and maintains graph connectivity. The borders of the cells define a subgraph referred to here as the *low resolution grid*. The following discussion analyzes the expected path degradation of such a spanner with $k = 4$.

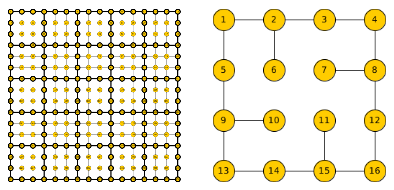

Figure 2: Left: Full grid with 4×4 cells. Right: the modified 4×4 cell.

Consider a square, 2-dim $n \times n$ lattice graph. Figure 2 (left) shows the 4×4 cells, where the low resolution grid is outlined. The construction replaces each of the 4×4 cells in the full grid with the modified cell depicted in Figure 2 (right). This structure bounds the distance from any node to the boundary by $\frac{k}{2}$, and is generalizable to arbitrary k. Examples for larger k can be seen in the next section.

Notice that any path on this spanner can be decomposed into sections that traverse the inside of at most two 4×4 cells and that traverse the low resolution grid. Searching for a path is hierarchical:

- First get onto the low resolution grid.
- Then find a path from the start cell to the goal cell.
- Lastly, depart the low resolution grid.

Our goal is to derive an expression of the form $\alpha \cdot d_G + \beta$ to approximate the expected path length of a path on the spanner given d_G, the path length of the path on the full grid. This can be achieved by approximating the average multiplicative stretch on the low resolution grid for α and providing a worst case β to account for the sections of the path that need to enter the 4×4 cells.

Sparseness of the structure: First, we compute the number of edges that are pruned by this construction to show the reduction of the search space size. The proposed spanner retains only $\frac{5}{9}$ of the full grid's edges, i.e., already close to the number of edges of a spanning tree for large numbers of vertices. Further analysis shows that only $\frac{5}{9}$ of the nodes are considered during a search on the spanner.

The number of nodes can be reduced further by understanding that nodes between intersection points on the low resolution grid only need to be considered in a search if one of those nodes is the goal. The path has no choice at these nodes but to proceed in one direction. If we only count intersection nodes on the low resolution grid, we find that the search space is reduced by a factor of $\frac{1}{9}$.

Approximating expected degradation: To compute the average multiplicative stretch in the low resolution grid, it is necessary to (a) first find the average degradation among all paths with degradation, (b) count the number of paths with degradation, and then (c) calculate a weighted average among all paths with degradation and paths without degradation. The analysis follows from the fact that any path with degradation must exist within a row or column of 4×4 cells. An approximation for the average multiplicative degradation on the low resolution grid has the following form:

$$\alpha = 1 + \mathcal{O}(\frac{\log(n) + n}{n^2}) \qquad (1)$$

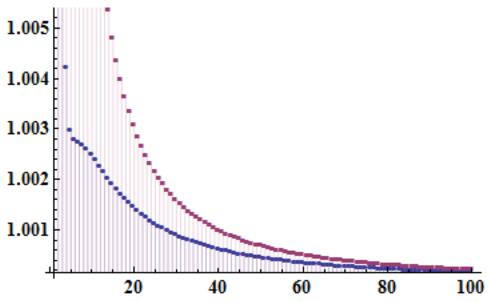

Figure 3: Average Degradation vs p, the number of cells in a row of cells. The blue line is the approximation.

This is an under-approximation but it is close to the real average on the low resolution grid. Figure 3 shows how the approximation compares to the actual average degradation on the low resolution grid. Notice that the worst case degradation is $\frac{5}{3} \approx 1.667$, which is significantly larger than the scale of the plot. The additive term incurred by traversing the cells is at most k, ($k = 4$ in the 4×4 case). Then, for the 4×4 case, we have: $(\forall_{u,v \in V})\ d_H(u, v) = \alpha \cdot d_G(u, v) + 4$, where α comes from Eq. 1.

General Case

The construction can handle obstacles by bordering them with edges, i.e., if a path encounters an obstacle, it can move along its boundary. Experiments show that path degradation increases slightly for low obstacle densities but then significantly decreases for higher densities as expected at the expense of the relative sparseness (Figure 4).

Let the average degradation be $\frac{a}{b}$, where a is the sum of the degradations of all paths, and b is the number of paths. Any obstacle will eliminate paths. In the worst case, all paths to and from the blocked node have no degradation.

Since the average degradation must be at least 1, $\frac{a}{b} \leq \frac{a-n^2}{b-n^2}$, the new average degradation after adding the obstacle is larger than the average degradation without the obstacle, which increases average degradation with obstacles.

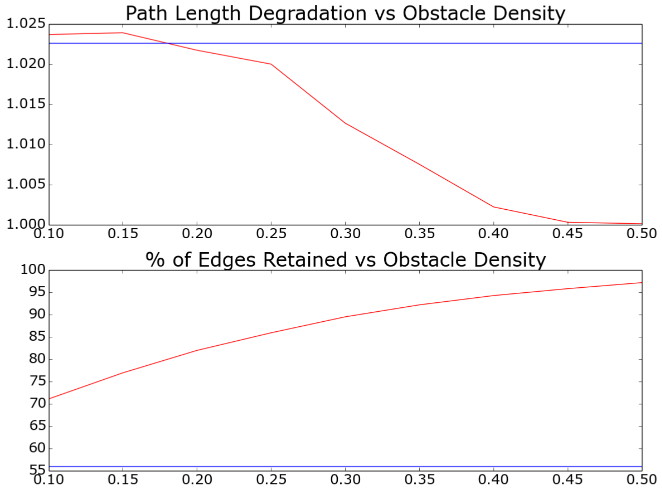

Figure 4: Data for a spanner with 4×4 cell size on a 100×100 grid. Straight lines: obstacle free case. Degradation greater than 1.0 indicates longer paths.

Obstacles can also block paths that are unique on the spanner, but not unique on the full grid; if the path is unique on the spanner, blocking it will require a longer detour, while the length on the grid stays the same. However, since we are bounding obstacles with edges, the average degradation quickly drops with increased obstacle density. The number of retained edges significantly increases with higher obstacle density, implying the construction is more beneficial in search spaces with few obstacles.

From 4×4 cells to $k \times k$ cells: As k gets larger, the average degradation on the low resolution grid decreases; however, the additive term increases. Furthermore, with larger k, there is less redundancy in paths in the low resolution grid. Thus, obstacles will affect path degradation more than they do in the 4×4 case. We expect to see a slow decrease in the number of edges pruned; the 4×4 case prunes $\frac{4}{9}$ of the edges for large n: close to the limit for retaining connectivity of $\frac{1}{2}$.

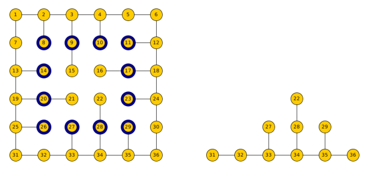

Figure 5: Left: A 6×6 cell with a 4×4 cell embedded in the center. Right: The pattern associated with any boundary of the 6×6 cell.

Figure 5 depicts the generalization. The structure in Figure 5 (right) compose the 6×6 cell. Cells of any size can be decomposed into a similar, easily determined pattern.

Experiments

This construction was implemented given the HOG2 repository using commercial maps from StarCraft, Warcraft 3, Baldurs Gate, and Dragon Age (Sturtevant 2012). Experiments were conducted also with room maps, mazes, and maps with random obstacles. They called A* search on the full grid and on the proposed spanner.

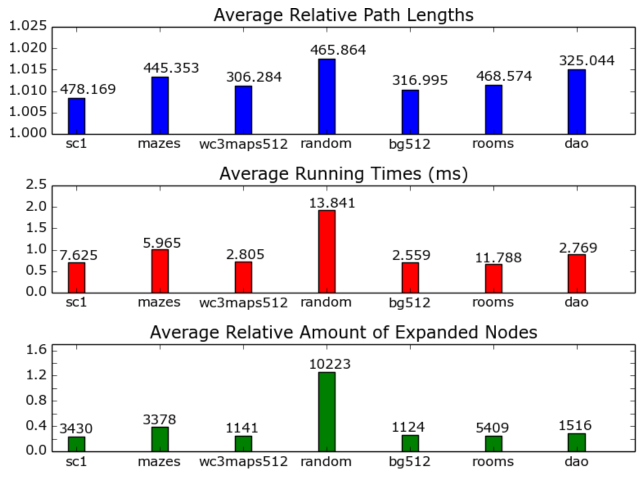

Figure 6: Data for 4×4 cells presented as an average for paths of each category. The bar heights represent relative results, and the values on top of the bars are average absolute results. Runtimes are given in milliseconds.

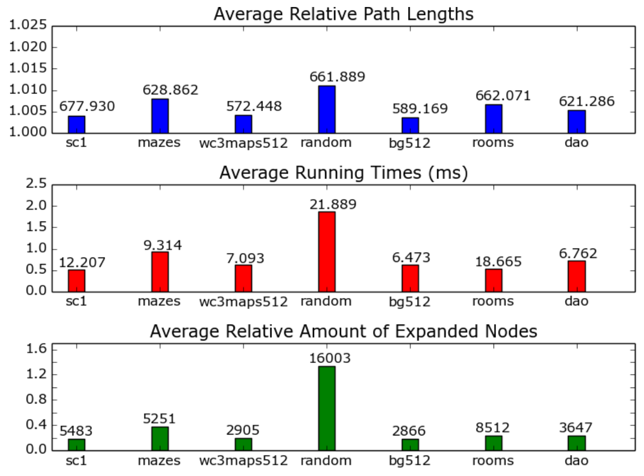

Figure 7: Same data set as Figure 6, excluding paths with lengths less than roughly 500 units

Path length, runtime, and number of expanded nodes are the metrics considered in the evaluation of the proposed spanner. For each map, we performed measurements for at most 73 paths using a variety of path lengths. The goal of using the spanner is to reduce query times by reducing the size of the search space, while keeping the path length as similar to the optimal path as possible. Measuring path lengths show how close to optimal the computed paths are. Measuring the number of expanded nodes provides an idea at how well the spanner reduces the search space, and measuring the run times illustrates how well the spanner performs in practice.

To best work with the existing infrastructure in the repository, we implemented this construction by modifying the A* implementation, such that it only considers nodes that would be in our spanner. Our implementation attempts to minimize the number of expanded nodes in a search. One way to do this is by searching strictly in a hierarchy. First, a path is found from the start point to the low resolution grid. Then, a path is found from the end point to the low resolution grid. Finally, a path is computed on the low resolution grid. When planning a path on the low resolution grid, nodes that are within a cell are not expanded, unless an obstacle on the low resolution grid makes it necessary. Thus, while the optimal path might exist on the spanner for obstacle-dense maps, this search algorithm might not find it, because the optimal path might involve cutting through cells to take shortcuts induced by obstacles. Furthermore, we do not expand nodes between intersection points on the low resolution grid, unless there is an obstacle or goal between the two intersection points. Since we are not expanding nodes within cells, there is no reason to expand the nodes between intersection points if there is no obstacle or goal between them.

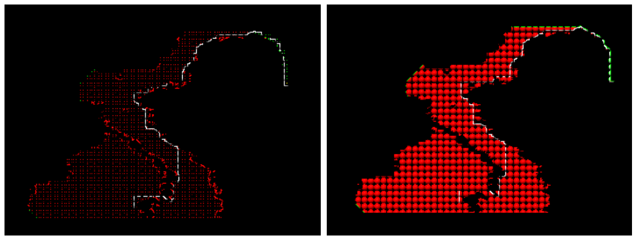
Figure 8: Left: search space with our spanner. Right: search space on full grid. Path stretch is 1.0047. White line indicates the found path.

The construction significantly reduced the size of the search space, improving running time for maps with large open areas. Our construction performed poorly on maps with random obstacle distribution, as expected. The optimal path might involve taking shortcuts constructed by obstacles, which our search algorithm does not consider in order to significantly reduce the number of nodes expanded in instances where the existence of such shortcuts is unlikely.

We found that gains in running time were clearer when only looking at data for longer paths (paths on the order of 500 units in length or longer). For extremely small paths, such as ones that start and end within a cell, we expect to see increases in running time. If a path starts and ends in the same cell, then the path has to first leave the cell, traverse the low resolution grid, and re-enter the cell. For slightly longer, but still small paths, running times for regular A* on the full grid is on the order of tens of microseconds. In these cases, the additional logic involved in our implementation to avoid expanding nodes not part of our spanner is likely to increase run times. While the ratios of the expanded nodes might be significant, the actual difference in number of expanded nodes is likely not enough in shorter paths to offset the cost incurred by additional logic.

Figure 8 shows the difference in the searched space between our construction and the full grid on maps with sections of low obstacle density. This figure shows the spanner with 4×4 cell width.

Fig 9 shows how the spanner performs on a map with randomly placed obstacles. In this case, the spanner is still using a 4×4 cell size, and the map has 10% obstacle density.

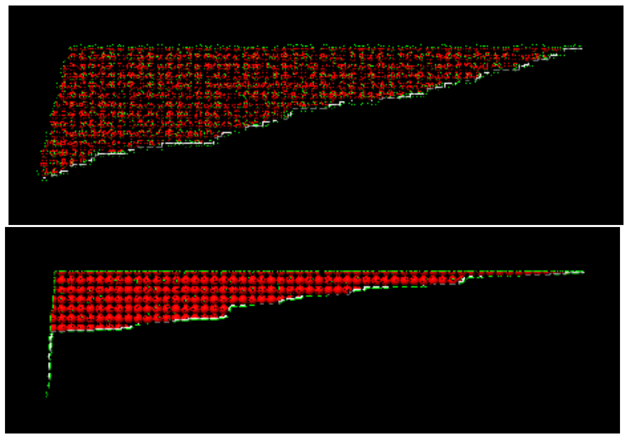
Figure 9: Top: search space with our spanner. Bottom: search space on full grid. Path stretch is 1. White line indicates the found path.

Discussion

This work proposes a new spanner construction for grids that works well on maps that have areas with few obstacles. For the case with no obstacles, theoretical analysis for a spanner with 4×4 cell size shows that the spanner is sparse, meaning that $\frac{4}{9}$ of the edges are pruned, while having low expected path degradation. This means that for large maps, the length of a path on the spanner is expected to be within 4 units of 1.002 times the path length on the full grid. For maps with open areas, the spanner significantly reduces query times and the number of nodes expanded in a search.

While experiments show where the method succeeds, it also illustrates where it is lacking. The spanner performs poorly on mazes and maps with random obstacle distribution; running times can increase on maps with random obstacle distribution. Ideally, maps using the proposed approach in practice would make sure during the preprocessing step to only use it in open areas. For maps with open areas, performance gains are seen for the vast majority of paths and are more prominent for longer paths.

One direction for future research is generalizing this construction to multiple dimensions. Multi-dimension path planning has applications in many areas, such as motion planning and manipulation. Furthermore, by performing the expected case analysis, this work is a first step for expected path degradation analysis on general graph spanners.

Acknowledgements

The authors are with the Computer Science Dept. at Rutgers University, NJ, USA. Their work is supported by NSF awards IIS-1451737, CCF-1330789, an RU-Aresty fellowship to Robert Kolchmeyer, and a DHS-CCICADA fellowship to Andrew Dobson. Any opinions, findings and conclusions or recommendations expressed in this paper do not necessarily reflect the views of the sponsors.

References

Baswana, S.; Kavitha, T.; Mehlhorn, K.; and Pettie, S. 2010. Additive spanners and (α, β)-spanners. *TALG* 7(1).

Botea, A.; Muller, M.; and Schaeffer, J. 2004. Near optimal hierarchical path-finding. *Journal of Game Development*.

Dobson, A., and Bekris, K. 2014. Sparse Roadmap Spanners for Asymptotically Near-Optimal Motion Planning. *IJRR* 33(1):18–47.

Dobson, A.; Krontiris, A.; and Bekris, K. 2013. Sparse Roadmap Spanners. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Elkin, M.; Emek, Y.; Spielman, D.; and Teng, S. 2008. Lower-stretch spanning trees. *SIAM Journal on Computing*.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *TSSC* 4(2):100–107.

Janssen, S. 2011. Improving heuristics for pathfinding in games. In *14th Twente Student Conference on IT*.

Koenig, S.; Likhachev, M.; and Furcy, D. 2004. Lifelong Planning A*. *Artificial Intelligence Journal* 155(1-2):93–146.

Liestman, A., and Shermer, T. 1993. Grid spanners. *Networks*.

Likhachev, M.; Gordon, G.; and Thrun, S. 2004. ARA*: Anytime A* with Provable Bounds on Sub-Optimality. NIPS.

Marble, J. D., and Bekris, K. 2011. Asymptotically near-optimal is good enough for motion planning. In *International Symposium on Robotics Research*.

Marble, J., and Bekris, K. 2013. Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE Transactions on Robotics* 29(2):432–444.

Peleg, D., and Schäffer, A. 1989. Graph Spanners. *Journal of Graph Theory* 13:99–116.

Pochter, N.; Zohar, A.; Rosenschein, J.; and Felner, A. 2010. Search space reduction using swamp hierarchies. In *Proceedings of the Third Annual Symposium on Combinatorial Search (SOCS-10)*.

Salzman, O.; Shaharabani, D.; Agarwal, P. K.; and Halperin, D. 2014. Sparsification of motion-planning roadmaps by edge contraction. *International Journal of Robotics Research* 33(14):1711–1725.

Shaharabani, D.; Salzman, O.; Agarwal, P. K.; and Halperin, D. 2013. Sparsification of Motion-Planning Roadmaps by Edge Contraction. In *Proc. of the IEEE Intern. Conf. on Robotics and Automation (ICRA)*, 4083–4090.

Storandt, S. 2013. Contraction hierarchies on grid graphs. In *KI 2013: Advances in Artificial Intelligence*.

Sturtevant, N. R., and Buro, M. 2005. Partial pathfinding using map abstraction and refinement. In *Proc. 20th Nat. Conf. Artif. Intell.*

Sturtevant, N. 2012. Benchmarks for Grid-Based Pathfinding. *T-CIAIG* 4(2):144–148.

Thayer, J. T., and Ruml, W. 2011. Tutorial: A Survey of Suboptimal Search Algorithms. In *Twenty-first International Conference on Automated Planning and Scheduling (ICAPS-11)*.

Uras, T., and Koenig, S. 2014. Identifying hierarchies for fast optimal search. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Wang, W.; Balkcom, D.; and Chakrabarti, A. 2013. A Fast Streaming Spanner Algorithm for Incrementally Constructing Sparse Roadmaps. IROS.