Speeding-up Any-Angle Path-Planning on Grids

Tansel Uras and **Sven Koenig**
Department of Computer Science
University of Southern California
Los Angeles, USA
{turas, skoenig}@usc.edu

Introduction

Search using Subgoal Graphs was a non-dominated approach in Grid-Based Path-Planning Competitions 2012 and 2013. In this paper, we take advantage of the similarities between Subgoal Graphs and visibility graphs to show that Subgoal Graphs can be used, with small modifications, to quickly find *any-angle paths*, thus extending their applicability. An any-angle path on a grid is a sequence of grid corners, where consecutive pairs of corners have line-of-sight (that is, the straight line between them does not pass through the interiors of blocked cells). Since movement along any-angle paths is not constrained to grid edges, any-angle paths are shorter and more realistic looking than shortest grid paths (paths that only use grid edges). We describe two variants of Subgoal Graphs, Simple Subgoal Graphs and N-Level Subgoal Graphs, and how they can be modified to find any-angle paths.

(Any-Angle) Simple Subgoal Graphs

Simple Subgoal Graphs (SSGs) (Uras, Koenig, and Hernández 2013) are constructed from grids whose vertices are placed at the centers of grid cells and can be used to find shortest grid paths. For the any-angle version, we move the vertices of the grid to the corners of grid cells, which does not change how SSGs are used.

We start with some definitions: A cell corner s is called a *subgoal* if and only if it is a convex corner of an obstacle (a contiguous set of blocked cells). Two cell corners s and u are called *h-reachable* if and only if there is a shortest grid path between them whose length is equal to the Octile distance (that is, the length of a shortest grid path assuming the grid has no blocked cells) between them. They are called *direct-h-reachable* if and only if they are h-reachable and none of the shortest grid paths between them pass through a subgoal (except for s and u).

Simple Subgoal Graphs are constructed by adding edges between all pairs of direct-h-reachable subgoals. The length of each edge is the Octile distance between the subgoals it connects. Figure 1 shows an example of an SSG. Observe that B3 and F5 are h-reachable but not direct-h-reachable (due to the subgoal at C4), so there is no edge between them.

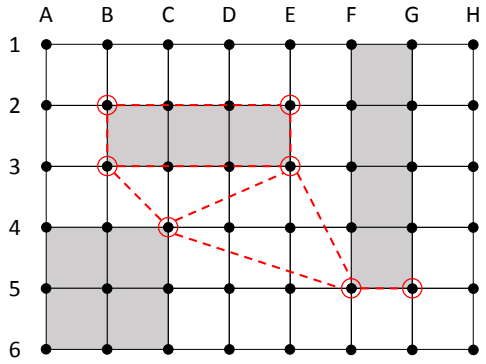


Figure 1: A Simple Subgoal Graph.

To find shortest grid paths using SSGs, one connects the given start and goal vertices s and g to their respective direct-h-reachable subgoals and searches this graph with A* to find a sequence of direct-h-reachable subgoals connecting s and g, called a *shortest high-level path*. One can then determine a shortest grid path between consecutive subgoals to find a shortest grid path between s and g. For instance, if we were to use the SSG in Figure 1 to find a shortest grid path between B1 and H3, we would connect B1 to subgoals A and B, H3 to subgoal F, and search this graph to find the shortest high-level path B1-D1-D3-F5-H5-H3. Following this high-level path on the grid, we obtain the shortest grid path B1-C1-D1-D2-D3-E4-F5-G5-H5-H4-H3.

Identifying direct-h-reachable subgoals from a given cell can be done efficiently with a dynamic programming algorithm that uses precomputed clearance values. Using this algorithm, SSGs can be constructed within milliseconds and the start and goal vertices can be connected to the SSGs quickly before a search.

To use SSGs to find any-angle paths, we first make the following observation: Any two corners that are direct-h-reachable also have line-of-sight (Uras and Koenig 2015b). This has two implications: 1) SSGs are sparser visibility graphs and therefore searching them can be significantly faster. 2) The high-level paths found by searching SSGs are any-angle paths, so we do not need to refine them into grid paths. We also use the Euclidean distance (rather than the Octile distance) as heuristic and as edge lengths to better guide the search into finding shorter any-angle paths (rather than shortest grid paths). We use Theta* (Daniel et al. 2010)

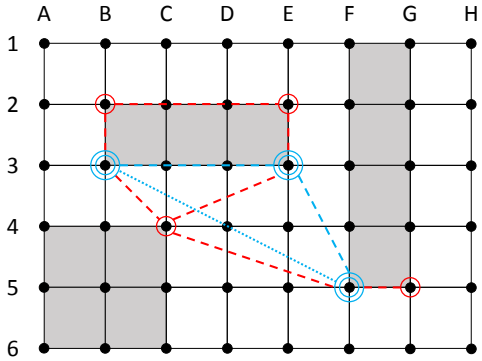
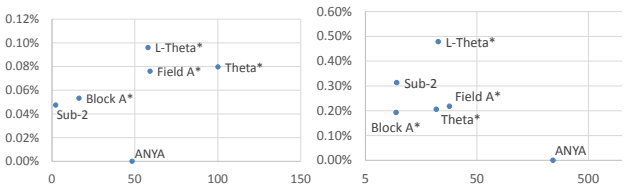


Figure 2: A Two-Level Subgoal Graph. Level 1 subgoals are shown in red, and level 2 subgoals are shown in blue.



(a) Game maps. (b) Random maps.

Figure 3: Path-length suboptimality (y-axis) and runtime (x-axis, in ms) after smoothing.

instead of A* to find even shorter any-angle paths.

(Any-Angle) N-Level Subgoal Graphs

N-Level Subgoal Graphs (Uras and Koenig 2014) are constructed from SSGs by creating a hierarchy among its vertices. This hierarchy is very similar to Contraction Hierarchies (Geisberger et al. 2008; Dibbelt, Strasser, and Wagner 2014), and can be used to exclude many vertices from the search (while maintaining optimality on grids), resulting in faster searches. In the resulting graph, each subgoal is assigned a level such that the following property holds for each level i: In the graph that contains exactly the subgoals of level i or higher (and the edges between them), the length of a shortest path between any two subgoals does not change if we remove any subset of level i subgoals (and their associated edges) from the graph.

During construction, one can add new edges to the graph (which allow searches to ignore even more vertices) to further improve runtime. Figure 2 shows a Two-Level Subgoal Graph constructed from the SSG in Figure 1 (by adding an edge between subgoals D and F). For brevity, we skip the details of how N-Level Graphs are constructed and searched.

In order to use N-Level Subgoal Graphs to find any-angle paths, we make the same changes that we do for SSGs. Furthermore, we restrict the construction of N-Level Subgoal Graphs to add new edges only between subgoals that have line-of-sight.

Experimental Results

The experiments are run on a PC with a dual-core 3.2GHz Intel Xeon CPU and 2GB of RAM. We use game maps and maps with randomly blocked cells in our comparison, which are available at Nathan Sturtevant's repository, along with the instances used for each map.[1] Figure 3 compares the runtime/path-suboptimality trade-offs of different any-angle path-planning algorithms, including the any-angle version of Two-Level Subgoal Graphs (Sub-2), on game maps and maps with randomly blocked cells. The paths found by all the algorithms except ANYA (which finds shortest any-angle paths on grids) are smoothed after the search (Botea, Müller, and Schaeffer 2004) and the results include smoothing time. The results show that, on game maps, Sub-2 dominates all the algorithms in the experiment except ANYA, in terms of the runtime/path-suboptimality trade-off. However, on random maps, it does not perform so well and is dominated by Block A*. More detailed results and a more detailed explanation for the experimental setup can be found in (Uras and Koenig 2015a). A more detailed comparison between different variants of Subgoal Graphs can be found in (Uras and Koenig 2015b).

Acknowledgments

Our research was supported by NSF under grant numbers 1409987 and 1319966. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies or the U.S. government.

References

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

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

Dibbelt, J.; Strasser, B.; and Wagner, D. 2014. Customizable contraction hierarchies. *arXiv preprint arXiv:1402.0402*.

Geisberger, R.; Sanders, P.; Schultes, D.; and Delling, D. 2008. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Proceedings of the International Conference on Experimental Algorithms*, 319–333.

Uras, T., and Koenig, S. 2014. Identifying hierarchies for fast optimal search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 878–884.

Uras, T., and Koenig, S. 2015a. An empirical comparison of any-angle path-planning algorithms. In *Proceedings of the 8th Annual Symposium on Combinatorial Search*. Code available at: http://idm-lab.org/anyangle.

Uras, T., and Koenig, S. 2015b. Speeding-up any-angle path-planning on grids. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling*.

Uras, T.; Koenig, S.; and Hernández, C. 2013. Subgoal graphs for optimal pathfinding in eight-neighbor grids. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling*.

[1] http://movingai.com/benchmarks/