Subgoal Graphs for Eight-Neighbor Gridworlds[*]

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

Carlos Hernández
Depto. de Ingeniería Informática
Univ. Católica de la Ssma. Concepción
Concepción, Chile
chernan@ucsc.cl

Abstract

We propose a method for preprocessing an eight-neighbor gridworld to generate a subgoal graph and a method for using this subgoal graph to find shortest paths faster than A*, by first finding high-level paths through subgoals and then shortest low-level paths between consecutive subgoals on the high-level path.

Preliminaries

We assume that an agent operates on eight-neighbor gridworlds with blocked and unblocked cells. Movement is from grid center to grid center, and it is assumed that objects moving have the same diameter as a grid cell. As a result, diagonal movement is only allowed when both associated cardinal directions are also unblocked. For example, an agent can move from A1 to B2 iff B2, A2 and B1 are all unblocked. We use octile distance as the heuristic function $h(s, s')$ that estimates the distance between cells s and s'. We say that s and s' are *h-reachable* iff there is a path of length $h(s, s')$ between them. We say that s and s' are *safe-h-reachable* iff they are h-reachable and every trajectory of length $h(s, s')$ between them is unblocked.

Simple Subgoal Graphs

Our simple subgoal graphs in gridworlds are similar to visibility graphs in Euclidean planes (Lozano-Pérez and Wesley 1979). We proceed as follows to construct a simple subgoal graph: First, we make the unblocked cells that are diagonally away from the convex corners of obstacles the subgoals. For example, if cells C3, C4 and D3 are blocked, the cells B2, B5, D5, E2 and E4 are the subgoals. Second, we add edges between all pairs of *directly-h-reachable* subgoals, which are h-reachable subgoals that are not h-reachable through another subgoal. For example, consider pairwise h-reachable subgoals s, s' and s''. If there is a shortest path that connects s and s' and passes through s'', we do not add the edge between s and s'. Figure 1(a) shows part of the resulting simple subgoal graph. Connecting only directly-h-reachable subgoals can significantly reduce the size and branching factor of simple subgoal graphs, which have the following two properties. First, the distances between subgoals in simple subgoal graphs are the same as the distances between them in the gridworld. Second, all edges connect subgoals that are directly-h-reachable.

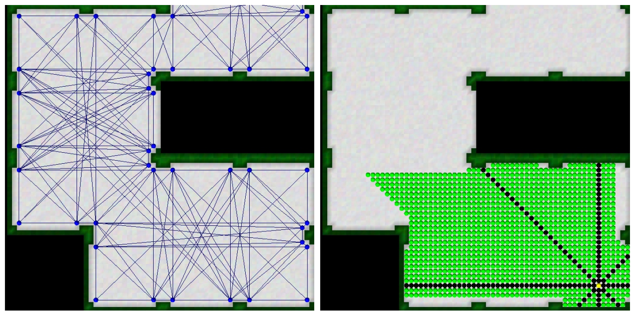
(a) Subgoals and directly-h-reachable edges (b) Directly-h-reachable area around the yellow cell

Figure 1: Simple subgoal graph

We proceed as follows to find a shortest path between a given start cell s and a given goal cell s': First, we make sure that s and s' are connected to the simple subgoal graph. If they are not subgoals, we add edges connecting them to all subgoals that they are directly-h-reachable with. Second, we perform an A* search (Hart, Nilsson, and Raphael 1968) from s to s'. It turns out that the length of the resulting high-level path is the same as the distance between s and s' in the gridworld. Third, we find a shortest path between s and s' in the gridworld by finding shortest paths between consecutive subgoals on the high-level path. It turns out that two subgoals that are directly-h-reachable are also safe-h-reachable, which means that any trajectory of length $h(s_i, s_{i+1})$ between subgoals s_i and s_{i+1} in the gridworld is unblocked and a shortest path between them can thus be constructed quickly.

An important part of constructing and searching the

[*]This paper is based upon research supported by NSF (while Sven Koenig was serving at NSF). It is also based upon research supported by ARL/ARO under contract/grant number W911NF-08-1-0468 and ONR in form of a MURI under contract/grant number N00014-09-1-1031. The research was performed while Carlos Hernández visited the University of Southern California.

simple subgoal graph is identifying all subgoals that are directly-h-reachable from a given cell s. We do this by exploring the directly-h-reachable area around s (shaded green, black and yellow in Figure 1(b)), which consists of all cells that are directly-h-reachable from s, and collecting all subgoals in this area.

Two-Level Subgoal Graphs

The runtime of the A* search in subgoal graphs dominates the runtime of connecting the start and goal cells to the subgoal graph and the runtime of finding low-level paths. We therefore speed up the A* search by reducing the size of the subgoal graph considered for the search, which is possible because some subgoals, called *local subgoals*, might only be needed to connect the start and goal cells to the subgoal graph. *Global subgoals* might also be required to connect the subgoals to each other. Our *two-level subgoal graphs* have the following two properties: First, the distances between subgoals through only global subgoals in two-level subgoal graphs are the same as the distances between them in the gridworld. Second, all edges connect subgoals that are h-reachable.

We proceed as follows to construct a two-level subgoal graph: First, we start with the simple subgoal graph and label all subgoals as global, at which point both properties are satisfied. Second, we iterate over all subgoals (in any order) and check if the current subgoal can be made local without violating the two properties. We label the current subgoal s as local iff s is not necessary to connect any pair of its neighboring subgoals s' and s'' in the current two-level subgoal graph, that is, iff (a) there is a path between s' and s'' through only global subgoals, excluding s, that is no longer than $h(s', s) + h(s, s'')$ (which means that there is a shortest path between s' and s'' that does not pass through s) or (b) s' and s'' are h-reachable (which means that there is a shortest path between s' and s'' that does not pass through s once we add an edge between s' and s''). We add edges between neighboring subgoals that satisfy (b) but not (a). The resulting two-level subgoal graph contains all edges of the simple subgoal graph plus the ones added during its construction. Whether a subgoal is labeled local or global might depend on the order in which the subgoals were iterated over. We call the subgraph of the two-level subgoal graph that consists of the global subgoals and the edges between them the *global subgoal graph*.

We proceed as follows to find a shortest path between a given start cell s and a given goal cell s': First, we make sure that s and s' are connected to the global subgoal graph. If they are not subgoals, we add edges connecting them to all local and global subgoals that they are directly-h-reachable with. We then add all local subgoals that are directly-h-reachable from the start or goal cells to the global subgoal graph and also their edges of the two-level subgoal graph from these local subgoals to the global subgoals and each other. Second, we perform an A* search from s to s'. It turns out that the length of the resulting high-level path is the same as the distance between s and s' in the gridworld. Third, we find a shortest path between s and s' in the gridworld by finding shortest paths between consecutive subgoals on the high-level path. This is more runtime intensive than in simple subgoal graphs since edges now connect h-reachable but not necessarily safe-h-reachable cells but can be done with breadth-first search or an A* search that does not generate cells with f-values that are larger than the octile distance between the two h-reachable subgoals.

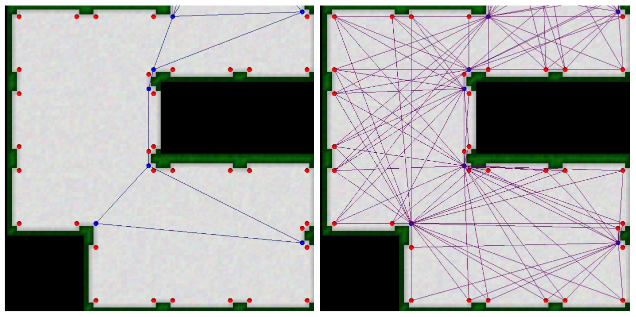
(a) Edges between two global subgoals (b) Edges between a global and a local subgoal

Figure 2: Two-level subgoal graph

Other Improvements

We also use three other improvements, whose specifics we omit: (1) We use pre-computed clearance values (namely, for each unblocked cell s and each cardinal move c, the number of moves that can be made from s in direction c before reaching a subgoal or blocked cell) to speed up the identification of the directly-h-reachable subgoals. We use the clearance values of directly-h-reachable cells that lie on the same vertical, horizontal or diagonal line as s (shaded black in Figure 1(b)) instead of exploring the directly-h-reachable area around s. (2) We discard all local-local edges (that is, edges between the local subgoals), which can be up to 70% of all edges, at the cost of not finding shortest paths on some rare occasions, which we mitigate somewhat at the cost of a slight increase in runtime. (3) We pre-compute the pairwise distances between all global subgoals and use them to find shortest paths without A* searches. For the competition, we use the two-level subgoal graph with pairwise distances but without local-local edges. If this is not possible within the given memory limit, we use the two-level subgoal graph without pairwise distances and local-local edges. If this is still not possible within the given memory limit, we perform regular A* searches with buckets.

References

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 2:100–107.

Lozano-Pérez, T., and Wesley, M. A. 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *Communication of the ACM* 22(10):560–570.