The Hierarchy in Grid Graphs (Extended Abstract)

Sabine Storandt
Albert-Ludwigs-Universität Freiburg
79110 Freiburg, Germany
storandt@informatik.uni-freiburg.de

Abstract

Many speed-up techniques developed for accelerating the computation of shortest paths in road networks, like reach or contraction hierarchies, are based on the property that some streets are 'more important' than others, e.g. on long routes the usage of an interstate is almost inevitable. In grids there is no obvious hierarchy among the edges, especially if the costs are uniform. Nevertheless we will show that contraction hierarchies can be applied to grid graphs as well. We will point out interesting connections to speed-up techniques shaped for routing on grids, like swamp hierarchies and jump points, and provide experimental results for game maps, mazes, random grids and rooms.

Introduction

Efficient route planning in grid graphs is important in a wide range of application domains, e.g. robot path planning and in-game navigation. While many heuristics, like A*, provide relatively fast solutions on the fly, it might be worthwhile to allow some preprocessing to speed up query answering if the same grid map is used multiple times. For road networks state-of-the-art preprocessing techniques like contraction hierarchies (Geisberger et al. 2008) and transit nodes (Bast et al. 2007) enable shortest path computation in a few ms or even μs on graphs with millions of nodes and edges. But the structure of street graphs differs clearly from grids: Shortest paths are almost always unique and some edges (e.g. corresponding to highways and interstates) occur in significantly more optimal paths than others. Therefore it is not obvious that such speed-up techniques carry over to grid graphs. Nevertheless, in (Antsfeld et al. 2012) it was shown that the idea of transit node routing can be adapted to grid graphs, leading to a significantly improved performance on video game maps. In this paper, we will show that contraction hierarchies allow for faster path planning in grids as well.

Contraction Hierarchies (CH)

The basic idea behind CH is augmenting the graph with shortcuts that allow to save a lot of edge relaxations at query time. To that end, in a preprocessing phase nodes are sorted according to some notion of importance. Afterwards the nodes get contracted one by one in that order, while preserving all shortest path distances in the remaining graph by inserting additional edges (so called shortcuts). More precisely, after removing a node v the distance between any pair of neighbours u, w of v has to stay unchanged. Therefore an edge (u, w) with proper costs is inserted if the only shortest path from u to w is uvw. If there exists a path with cost less than the ones of u, v, w – a so called *witness path* (typically found via a Dijkstra run from u) – the shortcut can be omitted. After all nodes have been removed, a new graph G' is created by adding all shortcuts to the original graph. An edge (v, w) is called upwards if the importance l of v is smaller than that of w ($l(v) < l(w)$) and downwards otherwise. In G' a shortest path can be subdivided into an upward and a downward path. Therefore s-t-queries can be answered bidirectionally, with the forward run (starting at s) considering only upward edges and the backward run (starting at t) considering exclusively downward edges. We call the respective subgraphs containing all upwards paths starting at s/ all downward paths ending in t as $G^{\uparrow}(s)/G^{\downarrow}(t)$ and the highest node wrt to l on an s-t-path the peak.

Adaption to Grids. On the first sight, the construction of a CH upon a grid seems to be a bad idea. Consider a 4-connected grid, the contraction of a node would remove four edges, but any two of these edges form a shortest path, hence six shortcuts must be inserted. Contracting the neighbouring nodes this effect amplifies, giving the impression that we might end up with a quadratic number of edges in G'. But there are two characteristics of grids preventing this: First, optimal paths are ambiguous, but only one optimal solution needs to be preserved. In fact, contracting the first node in a complete 4-connected grid, only two shortcuts instead of six have to be inserted because of ambiguity. Secondly, if the grid is far from being complete, the holes introduce a certain kind of hierarchy as now shortest paths tend to use their borders. Hence CH construction upon a grid might work after all.

There are no modifications necessary to run CH on a grid, but the preprocessing can be accelerated. Knowing the positions of the nodes in the grid, we do not have to start a witness search for u, v, w if the nodes are all in one row/column and the summed costs of $\{u, v\}$ and $\{v, w\}$ comply with the interval between u and w. Also if latter is true but not the row/column-property, we can restrict the witness search to

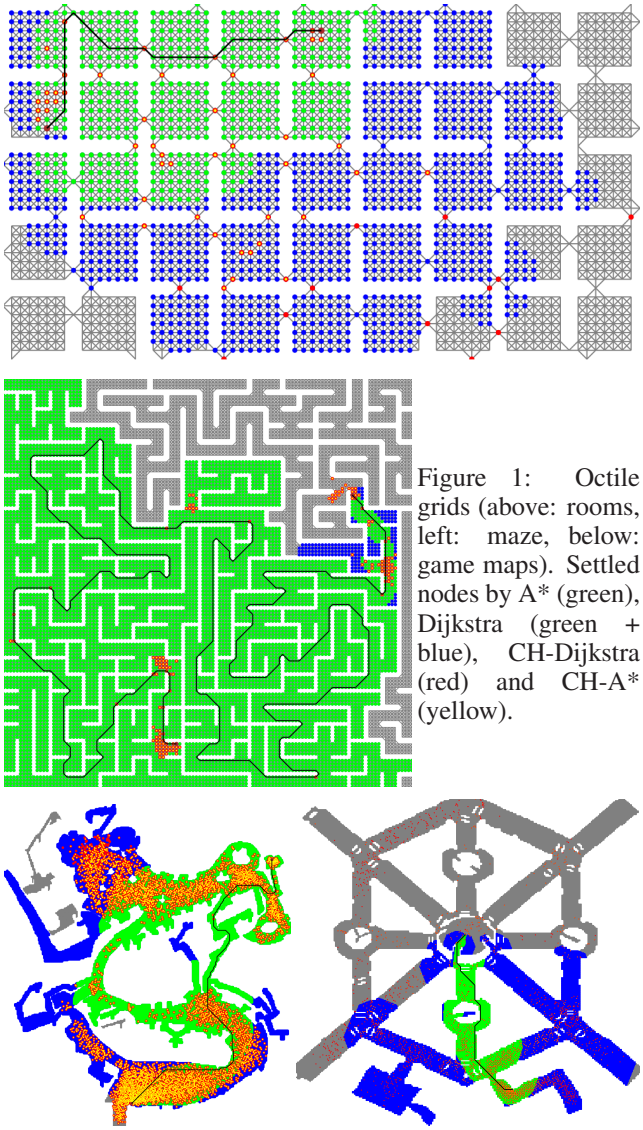

Figure 1: Octile grids (above: rooms, left: maze, below: game maps). Settled nodes by A* (green), Dijkstra (green + blue), CH-Dijkstra (red) and CH-A* (yellow).

the rectangle spanned by u and w. Moreover we can plug-in A* to accelerate the witness search in any case.

Maintaining Canonical Paths. To save energy and to enable a natural way of moving (especially for robots), we aim for an optimal path with the minimal number of turns, i.e. a canonical path. But neither plain Dijkstra nor A* can guarantee to find such a solution if the optimal path is ambiguous. To gain this ability in G', we assign classifiers to the shortcuts, indicating if the spanned path is pure horizontal or vertical, has a turn in the upper left, upper right, lower left or lower right corner, or is not trivially canonical (i.e. more than one turn). Bridging two edges via a new shortcut the classifier can easily be determined. As soon as a path u, v, w is trivially canonical, it has to be optimal and we insert the respective shortcut despite the possible existence of a witness.

Lemma 1 *Every trivially canonical s-t-path can be reconstructed considering only* $G^{\uparrow}(s) \cup G^{\downarrow}(t)$.

Proof. Let z be the turn point on the path. As s-z and t-z are unique optimal paths, they can be given in CH-description. So let p_1, p_2 be the peak nodes on those subpaths, w.l.o.g. $l(p_1) > l(p_2)$ and w the lowest node on the path p_1-z with $l(w) > l(p_2)$. As w-z goes downwards and z-p_2 upwards, the shortcut $\{w, p_2\}$ will be considered and inserted because it represents a canonical path. Hence s-p_1 is in $G \uparrow (s)$ and p_1-w-p_2-t in $G^{\downarrow}(t)$. ■

This can be extended to non-trivial canonical paths and octile grids.

Connections to other Speed-Up Techniques. In (Pochter et al. 2010) the concept of swamp hierarchies was introduced, a swamp being a set of nodes that can be excluded a priori from the search space for given s, t. In a CH-graph also the nodes $\notin G^{\uparrow}(s) \cup G^{\downarrow}(t)$ are pruned directly. Nevertheless the kind of blocked nodes differs, hence a combination of both approaches promises further improvement. The idea of jump points was presented in (Harabor and Grastien 2011) with a jump point being an 'important' point which has to be explored in the search process. Here no preprocessing is necessary, but sets of nodes between on the fly computed jump points are removed from the search space. The pruning rules applied there are similar to what happens during the CH-construction and they also compute canonical paths, hence it appeals that CH can be seen as an offline jump points approach.

Experiments

We evaluated the performance of the CH-Dijkstra and CH combined with A* as the average ratio of settled nodes by A* and nodes settled by our approachs on several instances extracted from http://movingai.com/, see the following table:

	mazes	rooms	game maps	random
CH-Dijkstra	52	105	4	10
CH-A*	51	120	6	15

We observe a reduction for all inputs. Interesting outcomes are that the number of settled nodes in mazes is often below the optimal path size, and that for rooms instances the door nodes were naturally declared important in the CH-construction. For game maps we observed mixed results, as some inputs responded better than others. See Figure 1 for some illustrations.

References

Antsfeld, L.; Harabor, D. D.; Kilby, P.; and Walsh, T. 2012. Transit routing on video game maps. In *AIIDE*.

Bast, H.; Funke, S.; Matijevic, D.; Sanders, P.; and Schultes, D. 2007. In transit to constant time shortest-path queries in road networks. In *ALENEX*.

Geisberger, R.; Sanders, P.; Schultes, D.; and Delling, D. 2008. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *WEA*, 319–333.

Harabor, D. D., and Grastien, A. 2011. Online graph pruning for pathfinding on grid maps. In *AAAI*.

Pochter, N.; Zohar, A.; Rosenschein, J. S.; and Felner, A. 2010. Search space reduction using swamp hierarchies. In *AAAI*.