Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks*

Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling

Universität Karlsruhe (TH), 76128 Karlsruhe, Germany
{robert.geisberger,sanders,schultes,delling}@ira.uka.de

Abstract. We present a route planning technique solely based on the concept of node *contraction*. The nodes are first ordered by 'importance'. A hierarchy is then generated by iteratively *contracting* the least important node. Contracting a node v means replacing shortest paths going through v by *shortcuts*. We obtain a hierarchical query algorithm using bidirectional shortest-path search. The forward search uses only edges leading to more important nodes and the backward search uses only edges coming from more important nodes. For fastest routes in road networks, the graph remains very sparse throughout the contraction process using rather simple heuristics for ordering the nodes. We have five times lower query times than the best previous hierarchical Dijkstra-based speedup techniques and a *negative* space overhead, i.e., the data structure for distance computation needs *less* space than the input graph. CHs can be combined with many other route planning techniques, leading to improved performance for many-to-many routing, transit-node routing, goal-directed routing or mobile and dynamic scenarios.

1 Introduction

Planning optimal routes in road networks has recently attracted considerable interest in algorithm engineering because it is an important application that admits a lot of interesting algorithmic approaches. Many of these techniques exploit the *hierarchical* nature of road networks in some way or another.

Here we present a very simple approach to hierarchical routing. Assume the nodes of a weighted directed graph $G = (V, E)$ are numbered $1..n$ in order of ascending 'importance'. We now construct a hierarchy by *contracting* the nodes in this order. A node v is contracted by removing it from the network in such a way that shortest paths in the remaining *overlay graph* are preserved. This property is achieved by replacing paths of the form $\langle u, v, w\rangle$ by a *shortcut* edge $\langle u, w\rangle$. Note that the shortcut $\langle u, w\rangle$ is only required if $\langle u, v, w\rangle$ is the only shortest path from u to w.

* Partially supported by DFG grant SA 933/1-3, and by the Future and Emerging Technologies Unit of EC (IST priority – 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

C.C. McGeoch (Ed.): WEA 2008, LNCS 5038, pp. 319–333, 2008.

We shall view the contraction process as a way to add all discovered shortcuts to the edge set E. We obtain a *contraction hierarchy (CH)*. Section 2 gives more details.

In Section 3 we explain how the nodes are ordered. Although 'optimal' node ordering seems a quite difficult problem, already very simple local heuristics turn out to work quite well. The basic idea is to keep the nodes in a priority queue sorted by some estimate of how attractive it is to contract a node. The main ingredient of this heuristic estimate is the *edge difference*: The number of shortcuts introduced when contracting v minus the number of edges incident to v. The intuition behind this is that the contracted graph should have as few edges as possible. Even using only edge difference, quite good CHs are computed. However, further refinements are useful. In particular, it is important to contract nodes 'uniformly'.

For routing, we split the CH (V, E) into an *upward graph* $G_\uparrow := (V, E_\uparrow)$ with $E_\uparrow := \{(u, v) \in E : u < v\}$ and a *downward graph* $G_\downarrow := (V, E_\downarrow)$ *with* $E_\downarrow := \{(u, v) \in E : u > v\}$. For a shortest path query from s to t, we perform a modified bidirectional Dijkstra shortest path search, consisting of a forward search in $G_\uparrow$ and a backward search in $G_\downarrow$. If, and only if, there exists a shortest s-t-path in the original graph, then both search scopes eventually meet at a node v that has the highest order of all nodes in a shortest s-t-path. More details of the query algorithm are given in Section 4. Applications and refinements like dynamic routing (i.e., edge weights are allowed to change), many-to-many routing, and combinations with other speedup techniques can be found in Section 5. Section 6 shows that in many cases, we get significant improvements over previous techniques for large real world inputs. Lessons learned and possible future improvements are summarized in Section 7.

Related Work

Since there has recently been extensive work on speed-up techniques, we can only give a very abridged overview with emphasis on the directly related techniques beginning with the closest kin. For a more detailed overview we refer to [1,2]. CHs are an extreme case of the hierarchies in highway-node routing (HNR) [3,2] – every node defines its own level of the hierarchy. CHs are nevertheless a new approach in the sense that the node ordering and hierarchy construction algorithms used in [3,2] are only efficient for a small number of geometrically shrinking levels. We also give a faster and more space efficient query algorithm using $G_\uparrow$ and $G_\downarrow$.

The node ordering in highway-node routing uses levels computed by *highway hierarchies* (HHs) [4,5,2]. Our original motivation for CHs was to simplify HNR by obviating the need for another (more complicated) speedup technique (HHs) for node ordering. HHs are constructed by alternating between two subroutines: *Edge reduction* is a sophisticated and relatively costly routine that only keeps edges required 'in the middle' of 'long-distance' paths. *Node reduction* contracts nodes. In the original paper for undirected HHs [5], node reduction only contracted nodes of degrees one and two, i.e., it removed attached trees and multihop