Parallel Contraction Hierarchies Can Be Efficient and Scalable (Supplemental Material)

Zijin Wan
University of California, Riverside
zijin.wan@email.ucr.edu

Xiaojun Dong
University of California, Riverside
xdong038@ucr.edu

Letong Wang
University of California, Riverside
lwang323@ucr.edu

Enzuo Zhu
University of California, Davis
ezzhu@ucdavis.edu

Yan Gu
University of California, Riverside
ygu@cs.ucr.edu

Yihan Sun
University of California, Riverside
yihans@cs.ucr.edu

A Query on Contraction Hierarchies

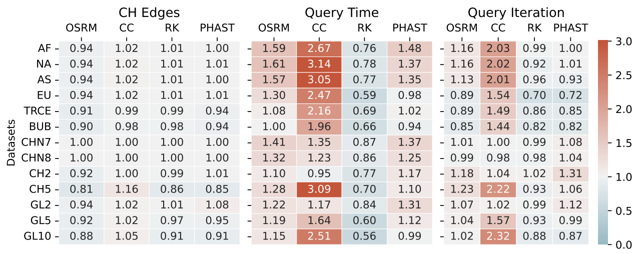
In the main paper, we introduce the construction algorithm for $G_{\mathrm{CH}} = (V, E_{\mathrm{CH}})$. In this section, we describe the standard query algorithm on a CH for completeness.

After constructing the CH, we re-index and sort all vertices by their level and score. In Geisberger's sequential CH algorithm [4], vertices are reordered based on the order to contract them. $\pi(v)$ is the order to contract v. In PHAST [2], vertices are re-grouped by levels to perform SSSP queries on CH, where vertices in lower levels always have a lower rank than those in higher levels. In the query phase, all the edges in E_{CH} are divided into two parts and forming two graphs: the out-edges from u to higher-ranked vertices are referred to as the ***upward edges***: ${E_{\mathrm{CH}}}^{\uparrow} \leftarrow \{(u,v) \mid \pi[u] < \pi[v]\}$, while the in-edges from higher-ranked vertices to u as the ***downward edges***: ${E_{\mathrm{CH}}}^{\downarrow} \leftarrow \{(v,u) \mid \pi[u] < \pi[v]\}$. For each edge connecting two vertices, only the endpoint of the lower rank stores this edge in G_{CH}.

CH boost up the efficiency of point to point query (s-t query) significantly. During a point-to-point query from s to t, a forward search from s is performed on ${E_{\mathrm{CH}}}^{\uparrow}$ and a backward search from t is performed on ${E_{\mathrm{CH}}}^{\downarrow}$, meaning that only edges leading to higher rank vertices are considered. For each vertex v, the distance from s to v and the distance from v to t is maintained as $d_s(v)$ and $d_t(v)$ and the estimated distance from s to t through v is $dist_s(v) + dist_t(v)$. Once the tentative shortest distance μ from s to v is not larger than the minimum value of the priority queue, the search result is settled to μ.

B More Experimental Results

Number of CH Edges, Query Time, and Query Iterations. To compare the number of CH edges, query time, and query iterations of our algorithm against the baselines (OSRM [5], CC [1], RK [3], PHAST [2]), we present a heatmap in Fig. 1. Here "query iterations" refers to the average number of vertices visited in an s-t query, which is machine-independent and roughly indicates the query cost. The numbers in the heatmap are normalized to those of our algorithm. For CH edges, the differences between the baselines and our algorithm are all within 20%, indicating that the space required to store the output CH is similar. For query time and iterations, RK and PHAST, being sequential and following a stricter contraction order, perform slightly better than the parallel implementations. In the worst case, our algorithm is only about twice as slow as the fastest query time. However, since queries finish in microseconds (10^{-6} seconds), this twofold slowdown is negligible. Among all parallel implementations, our algorithm achieves the best average query time and iterations. In summary, our algorithm offers significantly faster construction performance while maintaining competitive output graph size and query time.



Datasets	CH Edges				Query Time				Query Iteration			
	OSRM	CC	RK	PHAST	OSRM	CC	RK	PHAST	OSRM	CC	RK	PHAST
AF	0.94	1.02	1.01	1.00	1.59	2.67	0.76	1.48	1.16	2.03	0.99	1.00
NA	0.94	1.02	1.01	1.01	1.61	3.14	0.78	1.37	1.16	2.02	0.92	1.01
AS	0.94	1.02	1.01	1.00	1.57	3.05	0.77	1.35	1.13	2.01	0.96	0.93
EU	0.94	1.02	1.01	1.01	1.30	2.47	0.59	0.98	0.89	1.54	0.70	0.72
TRCE	0.91	0.99	0.99	0.94	1.08	2.16	0.69	1.02	0.89	1.49	0.86	0.85
BUB	0.90	0.98	0.98	0.94	1.00	1.96	0.66	0.94	0.85	1.44	0.82	0.82
CHN7	1.00	1.00	1.00	1.00	1.41	1.35	0.87	1.37	1.01	1.00	0.99	1.08
CHN8	1.00	1.00	1.00	1.00	1.32	1.23	0.86	1.25	0.99	0.98	0.98	1.04
CH2	0.92	1.00	0.99	1.01	1.10	0.95	0.77	1.17	1.18	1.04	1.02	1.31
CH5	0.81	1.16	0.86	0.85	1.28	3.09	0.70	1.10	1.23	2.22	0.93	1.06
GL2	0.94	1.02	1.01	1.08	1.22	1.17	0.84	1.31	1.07	1.02	0.99	1.12
GL5	0.92	1.02	0.97	0.95	1.19	1.64	0.60	1.12	1.04	1.57	0.93	0.99
GL10	0.88	1.05	0.91	0.91	1.15	2.51	0.56	0.99	1.02	2.32	0.88	0.87

Figure 1: Heatmap of number of CH edges, query time, and query iteration. Numbers are normalized to that of ours. Blue or Smaller is better.

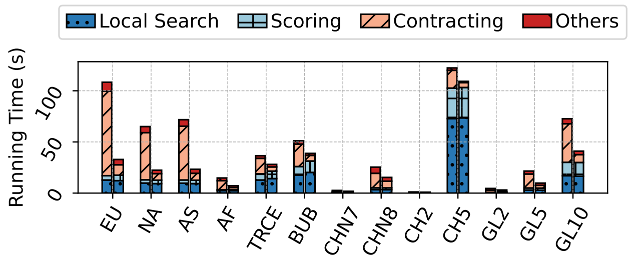


Figure 2: Running time with and without using the Lazy Combination Optimization. For each graph, the left bar represent the breakdown that combines in each round (without lazy combination), while the right bar represent that with the optimization. The cost is counted in the Contract step.

Evaluating the Lazy Combination of Shortcut Edges. As mentioned, one of our optimizations is to combine the shortcuts E^{+} lazily with the edges of the overlay graph E_O. This is to reduce and amortize the cost of updating the CSR for E_O. Fig. 2 shows the running time with and without this optimization. For each graph, the left bar represents the breakdown that combines in each round (without lazy combination), while the right bar represents that with the optimization. The cost is counted in the Contract step. By combining the shortcuts lazily instead of doing it every round, the cost of the Contract step is reduced from 40.2% to 10.4% of the overall time on average. Across all graphs, this optimization improves the performance by 1.1–3.3× for the total running time.

References

[1] Stefan Bühler and André Nusser. 2024. Contraction Hierarchies Constructor. https://doi.org/10.5281/zenodo.14008202

[2] Daniel Delling, Andrew V Goldberg, Andreas Nowatzyk, and Renato F Werneck. 2013. PHAST: Hardware-accelerated shortest path trees. *J. Parallel and Distrib. Comput.* 73, 7 (2013), 940–952.

[3] Julian Dibbelt, Ben Strasser, and Dorothea Wagner. 2016. Customizable contraction hierarchies. *J. Experimental Algorithmics* 21 (2016), 1–49.

[4] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. 2008. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *International Workshop on Experimental Algorithms (WEA)*. Springer, 319–333.

[5] Dennis Luxen and Christian Vetter. 2011. Real-time routing with OpenStreetMap data. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*. 513–516.