Contracting and Compressing Shortest Path Databases

Bojie Shen , Muhammad Aamir Cheema , Daniel D. Harabor , Peter J. Stuckey

Faculty of Information Technology, Monash University, Melbourne, Australia

Problem

Environments:

- Single agent navigation in road network.
- A road network $G = (V, E, w)$ is a graph, with nodes V, edges $E \subseteq V \times V$ and a weight function w maps each edge $e \in E$ to a non-negative weight $w(e)$.

Objectives:

- Find a shortest path from start to target.

Main Idea

Offline Preprocessing:
We build an oracle to find optimal paths: from any point to any other point in the graph. The oracle requires two ingredients:

- Contraction Hierarchy (CH) (see panel)
- CH-based Compressed Path Database (CH-CPD) (see panel)

Bi-directional CPD Search:
Given a partial CH-CPD constructed on a selected subset (top $n\%$) of the CH graph:

- We run two A* searches with landmark heuristic to incrementally find CH-CPD nodes for the start and target: v_s and v_t.
- We use oracle to build concrete paths, from start to target: $s \rightarrow v_s \rightarrow v_t \rightarrow t$.
- We keep the best path so far and eventually find the shortest path (sp).

Our approach is optimal and requires $|V_s| \times |V_t|$ path extractions. To improve the performance, we have two ingredients:

- Distance-based pruning (see panel)
- Cost caching pruning (see panel)

Advantages

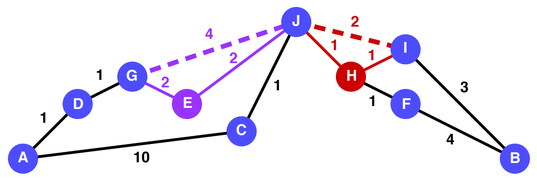
Preprocessing Cost:

- CH-CPD constructs faster than CPD.
- Partial CH-CPD reduces the preprocessing time and space further.

Query Performance:

- Full CH-CPD is faster than several state-of-art methods: CH; CH+L; CPD; and hub labeling methods: PHL, SHP.
- Partial CH-CPD tradeoff preprocessing time and space with query time, it is still highly competitive with other methods.

Contraction Hierarchy



A Contraction Hierarchy (CH) contracts the input graph G by adding shortcut. Apply a total lex order $\mathcal{L}$ (e.g., shown as alphabetical order) to the nodes of G. The CH contracts the graph as follow:

- W.r.t. $\mathcal{L}$, choose the least node v from the graph.
- Add to G a shortcut edge (u, w) between *each* pair of in-neighbour u and out-neighbour w of v for which: (i) $u >_{\mathcal{L}} v$ & $w >_{\mathcal{L}} v$; (ii) $v \in sp(u, w)$.

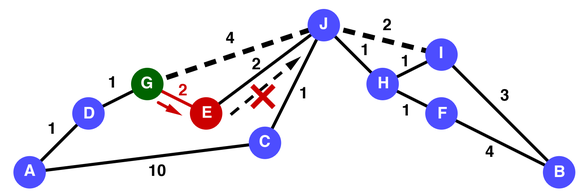
CH repeats the above steps until G is fully contracted (e.g., the result of contracting E (resp. H) is shown in purple (resp. red). Dashed edges indicate shortcut edges).

CH-based Compressed Path Databases

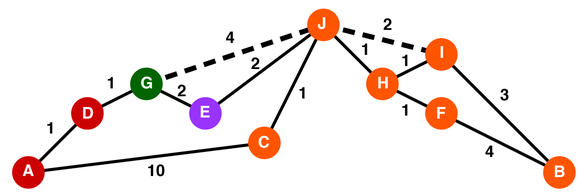
A CH-based Compressed Path Database (CH-CPD) is an all-pairs oracle that tells optimal first moves: from start to target. To build a CH-CPD, we run (i) modified Dijkstra search to compute first moves on the optimal CH paths, and (ii) distance table enhancement to improve the preprocessing time further.

- Modified Dijkstra Search:
  - Up-then-Down Policy. (i.e., only up; up-down; down paths are allowed).
- Distance Table Enhancement:
  - Compute first moves and cache the all-pairs distance for top $n\%$ of CH nodes.
  - Perform pruning with cached distance when compute first moves for other nodes.

The oracle needs to access a full APSP table but this is space prohibitive, so we compress with Run-Length Encoding (e.g., Row G compresses into four runs: 1D; 4J; 6E; 7J).



The Up-then-Down policy: the up successor J of E is pruned, because the predecessor G is lexically larger than E. (i.e., $\mathtt{G} >_{\mathcal{L}} \mathtt{E}$).



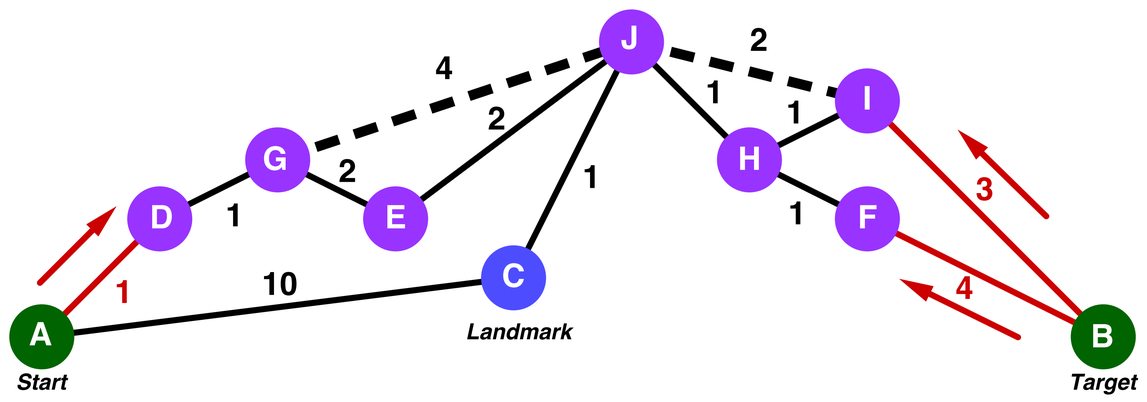
The source node G is highlighted as green. The first move on the optimal path from source node to any node are D, E and J shown as red, purple and orange, respectively.

Ordering	G	D	A	C	J	E	H	F	B	I
G	*	D	D	J	J	E	J	J	J	J
J	G	G	G	C	*	E	H	H	I	I
I	J	J	J	J	J	J	H	H	B	*

The symbol $[\mathtt{A} - \mathtt{J}]$ indicates the optimal first move, and $*$ is a wildcard symbol which can be compressed with any symbol.

Bi-directional CPD Search

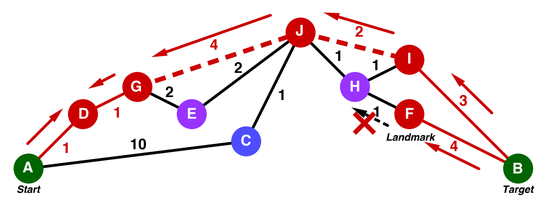
Ordering	G	D	A	C	J	E	H	F	B	I
$d(\mathtt{C}, \_)$	5	6	7	0	1	3	2	3	6	3



An example of bi-directional CPD search, where D to J are CH-CPD nodes (shown in purple) and C is a landmark (shown in blue). The A* search from start (resp. target) expands D (resp. I and F). C is pruned by landmark heuristic.

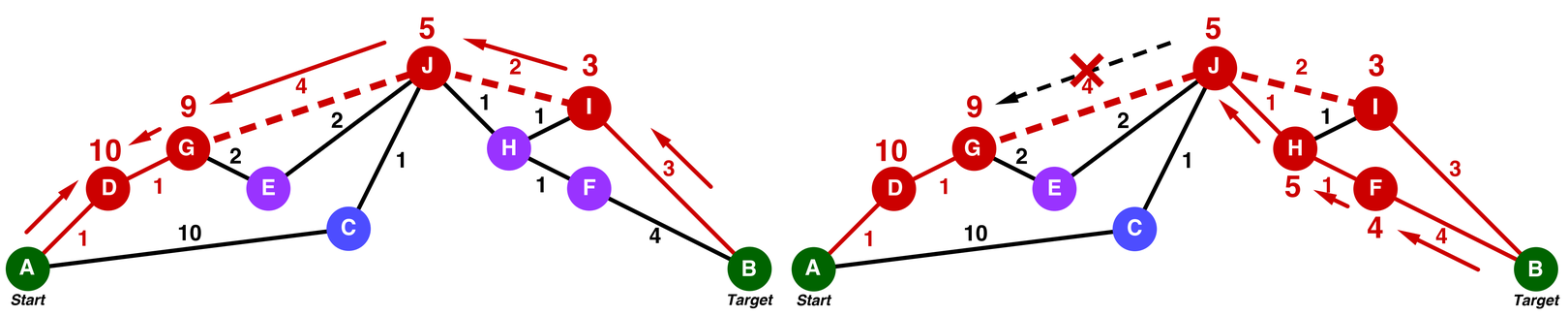
Distance-based Pruning

Ordering	G	D	A	C	J	E	H	F	B	I
$d(\mathtt{F}, \_)$	6	7	8	3	2	4	1	0	4	2



Avoid extracting the path from v_s to v_t, if $g(s, v_s) + landmark(v_s, v_t) + g(v_t, t) \geq |SP|$ (e.g., Assume F is a landmark, the path from F to D will not be extracted as $g(\mathtt{A}, \mathtt{D}) + landmark(\mathtt{D}, \mathtt{F}) + g(\mathtt{F}, \mathtt{B}) = 1 + |7 - 0| + 4 = 12 > 11$ (i.e., the cost of current SP: $|\langle \mathtt{B}, \mathtt{I}, \mathtt{J}, \mathtt{G}, \mathtt{D}, \mathtt{A} \rangle|$)).

Cost Caching Pruning



Bi-directional CPD search caches distance on each node when extracts the path from I to D. The path extraction from F to D terminates at J, because the cached distance $g(\mathtt{B}, \mathtt{J}) < g(\mathtt{B}, \mathtt{H}) + d(\mathtt{H}, \mathtt{J})$.

Experimental Results

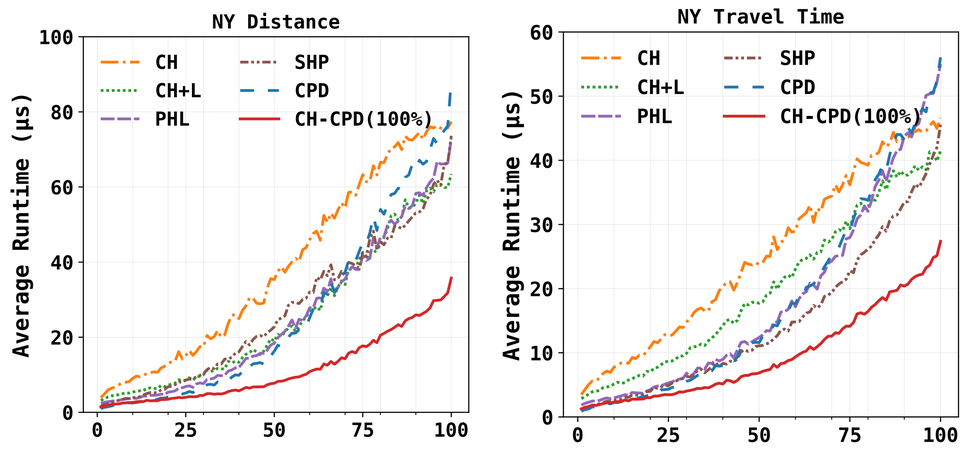
Benchmarks: We run experiments on a set of real-world road network using either the distance weights or the travel time weights, the results of New York (NY) dataset are shown as below:

Map Type	Preprocessing Cost																		
	Build Time (Mins)									Memory (MB)									
	CH-CPD					Competitors				CH-CPD					Competitors				
	20%	40%	60%	80%	100%	CPD	CH	PHL	SHP	20%	40%	60%	80%	100%	CPD	CH	PHL	SHP	
Distance	0.36	0.73	1.18	1.87	2.95	8.76	0.24	0.60	0.44	70	104	183	271	338	219	29	411	449	
Travel Time	0.27	0.96	1.79	2.24	3.00	11.03	0.16	0.18	0.17	63	88	156	222	277	188	28	161	198	

Build time in Mins, and memory cost in MB for CH-CPD and competitors

Map Type	Average Runtime (μs)									
	CH-CPD					Competitors				
	20%	40%	60%	80%	100%	CPD	CH	CH+L	PHL	SHP
Distance	23.17	20.83	17.19	13.67	11.42	26.38	38.64	25.58	25.36	26.76
Travel Time	19.16	18.82	14.83	12.85	9.53	18.23	25.27	20.06	18.32	14.93

Runtime comparison for shortest path queries, we report average time (µs) for CH-CPD and competitors.



Runtime comparison. The x-axis shows the percentile ranks of path queries sorted based on actual distances between start and target.