Efficient Route Planning on Public Transportation Networks: A Labelling Approach

‡ NANYANG TECHNOLOGICAL UNIVERSITY † Agency for Science, Technology and Research SINGAPORE § 復旦大學 FUDAN UNIVERSITY

Sibo Wang‡, Wenqing Lin†, Yi Yang§, Xiaokui Xiao‡, and Shuigeng Zhou§

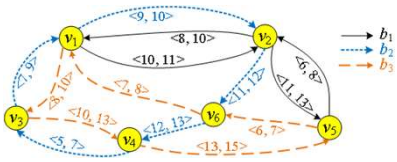
1. Problem Definition

- Public Transportation Network
  - Each edge $\langle u, v, t_d, t_a, b\rangle$ denotes that the bus b departures from station u at timestamp t_d and arrives at station v at timestamp t_a.

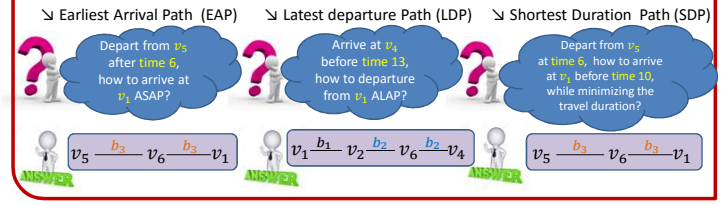
Bus b_1			
Dep. Station	Arr. Station	Dep. Time	Arr. Time
v_5	v_2	6	8
v_2	v_1	8	10
.....			

Bus b_2			
Dep. Station	Arr. Station	Dep. Time	Arr. Time
v_4	v_3	5	7
v_3	v_1	7	9
.....			

Bus b_3			
Dep. Station	Arr. Station	Dep. Time	Arr. Time
v_5	v_6	6	7
v_6	v_1	7	8
.....			

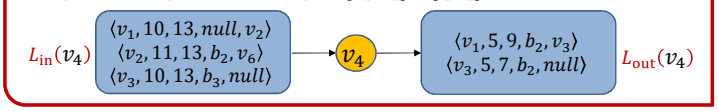


- Route Planning Queries
  - ↘ Earliest Arrival Path (EAP)
  - ↘ Latest departure Path (LDP)
  - ↘ Shortest Duration Path (SDP)



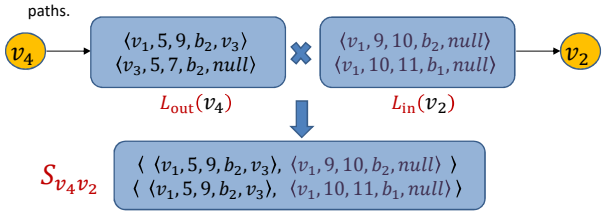
2. Timetable Labelling

- Node order o
  - A total ordering: for any u and v, we have $o(u) > o(v)$ or $o(u) < o(v)$.
  - u has a higher rank than v if $o(u) < o(v)$.
- Each node v has an in-label set $L_{\text{in}}(v)$ (resp. out-label set $L_{\text{out}}(v)$). Each label is a tuple $l = \langle x, t_d, t_a, b, p\rangle$, where
  - x: the staring node (resp. ending node) of a path P from x to v (resp. v to x). Among all nodes on P, x has the highest rank.
  - p: among all nodes on P except x and v, p is the node with the highest rank.
  - t_d, t_a, b: path P departures at time t_d, and arrives at time t_a, by taking bus b. If bus transfer is required, b is $null$.
  - No *dominated path* P' exists such that P' has the same starting node and ending node as P with departure time t'_d and arrival time t'_a, but $[t'_d, t'_a] \subseteq [t_d, t_a]$.

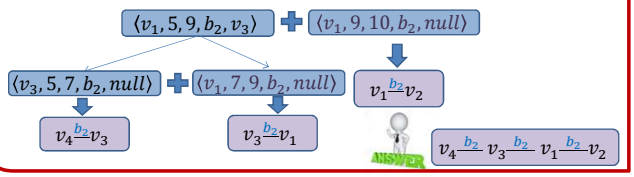


3. Query Processing

- Phase 1: Candidate Generation.
  - Consider the paths from u to v.
  - Given $l_1 = \langle w, t_d, t_a, \cdot, \cdot\rangle \in L_{out}(u)$ and $l_2 = \langle w, t'_d, t'_a, \cdot, \cdot\rangle \in L_{in}(v)$ where $t_a \le t'_d$.
    - Concatenating path pertinent to l_1 and l_2, add $\langle l_1, l_2\rangle$ to a candidate set S_{uv}.
  - Pruning techniques: exploit node ranks and time information to avoid dominated paths.

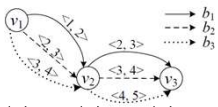


- Phase 2: Refinement and Path Reconstruction.



4. Preprocessing

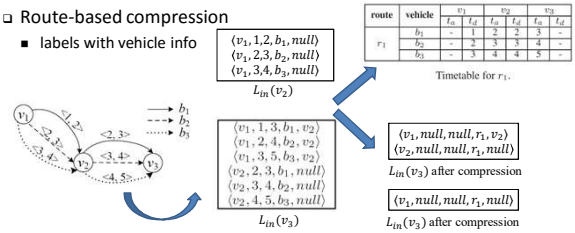
- Index Construction
  - Choose the node v with highest rank.
  - Forward EAP traversal.
    - From the largest timestamp t: non-dominance guarantee → P_t.
    - Next largest timestamp t': not dominated by P_t → non-dominance guarantee.
  - Backward LDP traversal, from smallest timestamp.
- Node Ordering
  - An $\sqrt{m}$ -Approximation Algorithm.
    - Adaptive coverage $R_i(v)$: # of non-dominated paths not covered by ordered vertices u_1, u_2, u_{i-1}
    - Node with maximum $R_i(v)$, we have $o(v) = i, i.e., u_i = v$.
    - Time complexity: $O(n^2 \cdot m)$, where n and m are numbers of nodes and edges respectively.
  - A Heuristic algorithm
    - Estimate adaptive coverage with sampling.



$R_1(v_1) = 6, R_1(v_2) = 9, R_1(v_3) = 6$;
Leading to $o(v_2) = 1$

5. Optimizations and Extensions

- Label Compression
  - Route-based compression
    - labels with vehicle info



route	vehicle	v_1		v_2		v_3	
		t_a	t_d	t_a	t_d	t_a	t_d
r_1	b_1	-	1	2	2	3	-
	b_2	-	2	3	3	4	-
	b_3	-	3	4	4	5	-

Timetable for r_1.

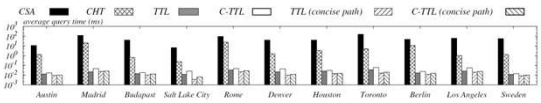
  - Pivot-based compression
    - $o(u) < o(p) < o(v)$
    - Consider $l_1, l_2, l_3 \in L_{in}(v)$.
      - $l_1 = \langle u, t_d, t_a, null, p\rangle$,
      - $l_2 = \langle u, t'_d, t'_a, null, p\rangle$,
      - $l_3 = \langle u, t_d, t^*_a, null, p\rangle$
    - Compress to $l_c = \langle u, null, null, null, p\rangle$
      - l_1, l_2, l_3 recover from $L_{in}(p), l_{in}(u)$
    - Label dependency can be used to build a weighted dependency graph.
    - NP-hard: maximum-weight independent set
- Concise Representation of Query Results
  - $P = \langle \langle v_3, v_1, 7, 9, b_2\rangle, \langle v_1, v_2, 9, 10, b_2\rangle, \langle v_2, v_5, 11, 13, b_1\rangle \rangle$
  - P can be denoted by a concise path P_c, as follows:
    - $P_c = \langle \langle v_3, 7, b_2\rangle, \langle v_2, 11, b_1\rangle \rangle$
  - Reduce path unfolding cost.
  - Improving query performance.

6. Experiments

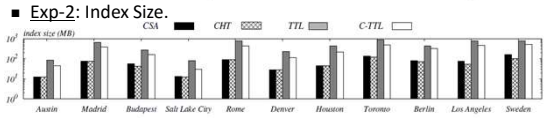
- Datasets

dataset	$\lvert V\rvert$	$\lvert E\rvert$	avg degree
Austin	2.7K	317.5K	118.7
Madrid	4.6K	1912.2K	412.5
Budapest	5.5K	1375.2K	251.5
Salt Lake City	6.3K	329.0K	52.6
Rome	8.8K	2267.7K	258.3
Denver	9.5K	708.7K	74.5
Houston	9.8K	1111.8K	112.9
Toronto	10.8K	3295.1K	305.4
Berlin	12.8K	1965.8K	153.0
Los Angeles	15.0K	1903.2K	126.6
Sweden	51.4K	3926.5K	76.4

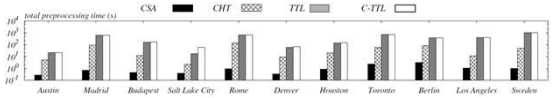
- Competitors
  - CSA [Dibbelt et al. SEA'13, Wu et al. VLDB'14]
  - CHT [Geisberger et al. SEA'10]
- Exp-1: Query Performance.



- Exp-2: Index Size.



- Exp-3: Preprocessing Time.



- Exp-4: Compression Ratios.
  - Δ_1: Route-based compression.
  - Δ_2: Pivot-based compression.
  - Δ_3: Route-based and Pivot-based compression.

dataset	$\Delta_1/\lvert\mathcal{L}\rvert$	$\Delta_2/\lvert\mathcal{L}\rvert$	$\Delta_3/\lvert\mathcal{L}\rvert$
Austin	20.25%	33.66%	46.20%
Madrid	16.52%	30.23%	45.61%
Budapest	10.41%	35.45%	41.71%
Salt Lake City	38.00%	38.78%	61.43%
Rome	14.42%	31.85%	43.87%
Denver	23.84%	34.92%	49.23%
Houston	28.77%	32.62%	49.54%
Toronto	19.95%	32.77%	44.78%
Berlin	6.70%	23.51%	27.58%
Los Angeles	16.78%	31.68%	41.66%
Sweden	12.74%	28.52%	35.92%