Frequency Data Compression for Public Transportation Network Algorithms

Hannah Bast **and** Sabine Storandt

FREQUENCY-BASED GRAPH CONSTRUCTION

Given: timetable information

Central Station	
ARRIVAL	DEPARTURE
B51 from Airport *9:35 10:05 10:35 11:05*	B17 to Harbour *9:45 10:00 10:15 10:30 10:45* travel time 8min
B82 from Hospital *9:43 9:51 9:59 10:07 10:15*	B82 to China Town *9:47 9:55 10:03 10:11 10:19* travel time 25min
B101 from Castle	B90 to Market S[illegible]

Goal: construct a space-efficient representation of the timetable information which allows for easy and fast route planning

Contribution: new graph model that takes advantage of the natural compressability of timetable information due to repeating service intervals

Approach: given a set of departure times, decompose them into a small set of *arithmetic progressions (APs)*, insert one edge for each AP

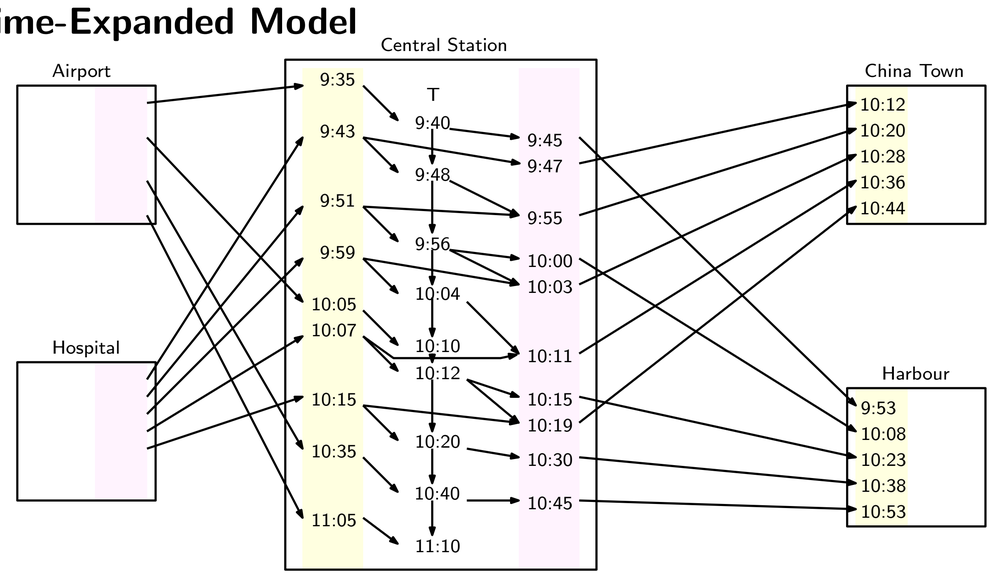
Example
7:00, 7:05, 7:11, 7:19, 7:25, 7:27, 7:30, 7:35, 7:45, 8:00, 8:05, 8:25, 8:30

- 7:00 - 8:30, every 30 minutes
- 7:05 - 8:25, every 20 minutes
- 7:11 - 7:35, every 8 minutes

Problem: optimal decomposition is NP-hard

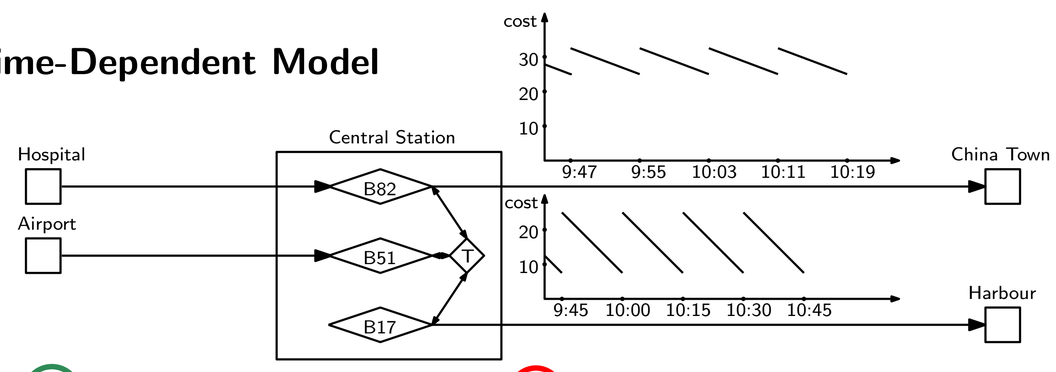
Remedy simple and fast heuristics achieve very good solutions in practice (provable via instance based lower bounds)

Time-Expanded Model



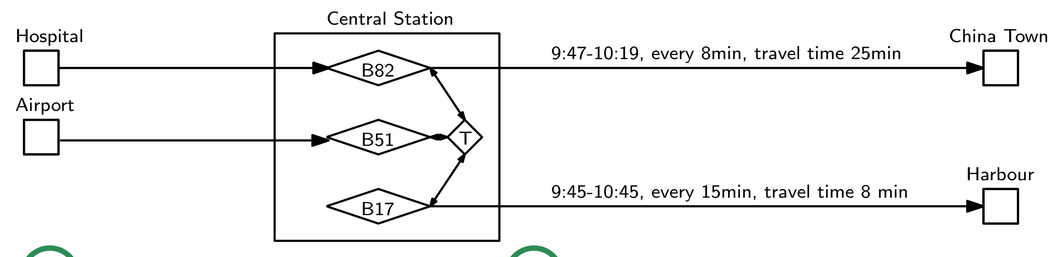
⊕ easy route planning ⊖ very space consuming

Time-Dependent Model

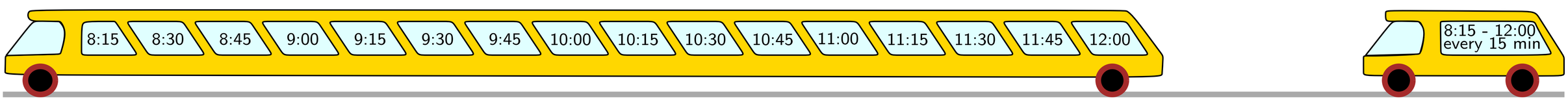


⊕ sparse graph ⊖ complex edge cost functions

NEW: Frequency-Based Model



⊕ sparse graph ⊕ constant sized edge labels



(PROFILE) QUERIES ON FREQUENCY DATA

Example

	SINGLE QUERY	PROFILE QUERY
H	@9:35	@ 9:00-13:00
	9:10-12:10 every 20min, travel time 8min	
H	@9:58	@ 9:18-12:18, every 20min, takes 8min
	9:20-12:20 every 20min, travel time 10min	
H	@10:10	@ 9:30-12:30, every 20min, takes 20min
	9:45-14:45 every 15min, travel time 5min	
H	@10:20	@ 9:50-12:50, every 60min, takes 40min @10:05-12:05, every 60min, takes 35min @10:20-12:20, every 60min, takes 30min
	9:54-14:54 every 15min, travel time 7min	
H	@10:31	@10:01-13:01, every 60min, takes 51min @10:16-12:16, every 60min, takes 46min @10:31-12:31, every 60min, takes 41min

Edge Evaluation

frequency-edge labels consists of

- a .... first service time
- b ... last service time
- f ... frequency
- c ... travel time

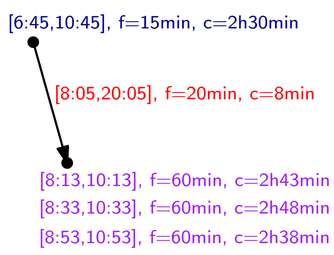
for single point in time t, frequency-edges can be evaluated in $O(1)$:

$$cost(t) = \begin{cases} a - t + c & \text{if } t < a \\ a + \lceil (t-a)/f \rceil \cdot f - t + c & \text{if } t \in [a, b] \\ \infty & \text{if } t > b \end{cases}$$

for profile queries Dijkstra assigns frequency-labels to the nodes with

- a ... first arrival time
- b ... last arrival time
- f ... frequency
- c ... summed travel time since departure from source

edge relaxation becomes a multi-step procedure:



$lcm(15, 20) = 60$
$start = 8:00$
$steps = 60/15 = 4$

dep	arr	cost
8:00	8:13	c=13min
8:15	8:33	c=18min
8:30	8:53	c=23min
8:45	8:53	c= 8min

Experimental Results

for the transit network of Madrid (4635 stations) we get a speed-up of

- 150 for **single queries**
- 10 for **profile queries**

compared to the time-expanded model