# Dynamic Time-Dependent Routing in Road Networks Through Sampling

Feb 2020

Motahhareh Nadimi

# Abstract

The earliest arrival in road networks with:

- TIME DEPENDENT functions
- DYNAMIC UPDATES
- It compute with SMALL ERROR
- Doesn't suffer from MEMORY LACK on large instances
- This algorithm is the only that is able to answer to queries below 50 ML

# Introduction

The road network is a weighted graph, directed graph :

Nodes: positions

Edges : road segments

Weight : travel time a car need to traverse the road segments .

A common assumption : travel times are time independent

Result: Traditional algorithms for finding such routes include Dijkstra's Algorithm and the A* algorithm.

Problem : road networks are huge and this algorithm is slow!

# Out line

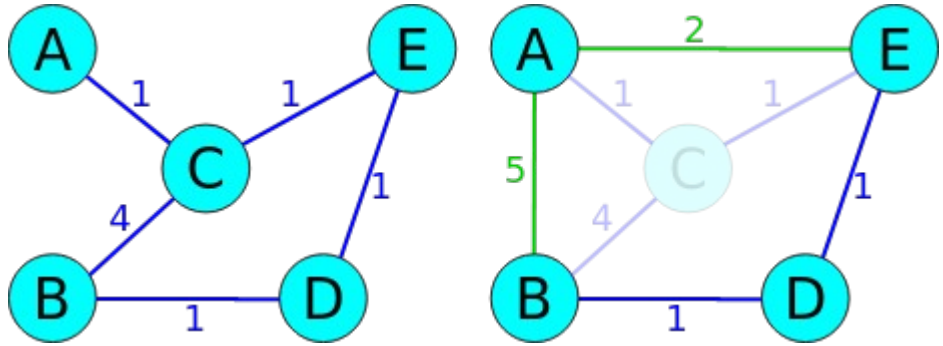
They start by describing a baseline that call the free flow heuristic.

Then build on it and introduce Their algorithms TD-S  and the profile extension TD-S+P and dynamic extension TD-S+D

# Recap contraction Hierarchies

Pre processing the graph by "contracting" the nodes one at a time(iterative vertex contractions)

When contracting a vertex V t is temporarily removed from the graph G, and a shortcut is created between each pair { u , w } of neighboring vertices if the shortest path from u to w contains v



# Recap on contraction Hierarchies

## Pre processing phase:

To calculate the distance between two cities, the algorithm has to traverse all the edges along the way, adding up their length. Pre computing the distance once and storing it in an additional edge created between the two large cities will save calculations each time this highway has to be evaluated in a query

## Finding the shortest path:

To find the shortest path between two nodes we perform two searches - one from the start node, one from the end node - then we see where they meet.

# Computing Errors

When taking predicted congestion into account, many proposed algorithms compute paths with an error

Let **d** denote the length of the computed path and **dopt** the length of a shortest path

Absolute error: $|d - dopt|$

Relative error : $|d - dopt| / dopt.$

# Computing Errors

Unfortunately, the predictions do not quantify **uncertainties**.

and uncertainty is huge: The time a typical German traffic light needs to cycle though its program is between 30s and 120s and Traffic lights often adapt to the actual real time traffic.

# PROBLEM DEFINITION

The input of earliest problem:

NODES > S AND T

Departure time > s

The task is consist of computing **ST- path with minimum arrival time.**

TD: Free Flow Heuristic: A solution to time-independent routing problem

TD-S : Solve the earliest arrival problem with **predicted congestion**

TD-S+P : solve **profile** problem

TD-S+D: take **real time congestion** into account

# TD routing algorithm :Free flow Heuristic

**The free flow travel time along an edge Is the minimum travel time over the whole day.**

| | | |
|---|---|---|
| **assumption** | **No Congestion (time-independent)** | The min value of e's travel **time function** of fe |
| **Step** | Find the **shortest** time independent **path H** | Compute **time** dependent travel time along **H given departure time (sampling)** |

**The first step can be fasten using any time independent speed up: such as CH**

**Avg flow heuristic** : the same but it use average travel time

# The Simplest TD Routing Algorithm: The Free flow Heuristic

Free flow never reroutes based on the current traffic situation

The main problem with it is that the computed paths can potentially differ significantly from the optimal ones.

The computed solutions can have a significant error.

# The Algorithm : TD-S

| | |
|---|---|
| **1** | Fix a small set of time intervals called time windows. Better k <=10 |
| **2** | For each time window and edge compute the **average travel time** in that window |
| **3** | For Each time window make a **time-independent graph.** |

It s time independent
Because we calculate avg

Pre process each of these graphs using your favorite time-independent speedup technique

IN STEP 2, THEY USE A SAMPLING APPROACH,
AVG THE TIME DEPENDENT TRAVEL TIME WITHIN EACH INTERVAL
FOR EX: 7:10, 7:15 , 7:50

# The query phase of TD-S

| | |
|---|---|
| 1 | For every graph they compute a shortest time independent path St-path and mark the edges in the path. |
| 2 | search several times for departure times at regular intervals |

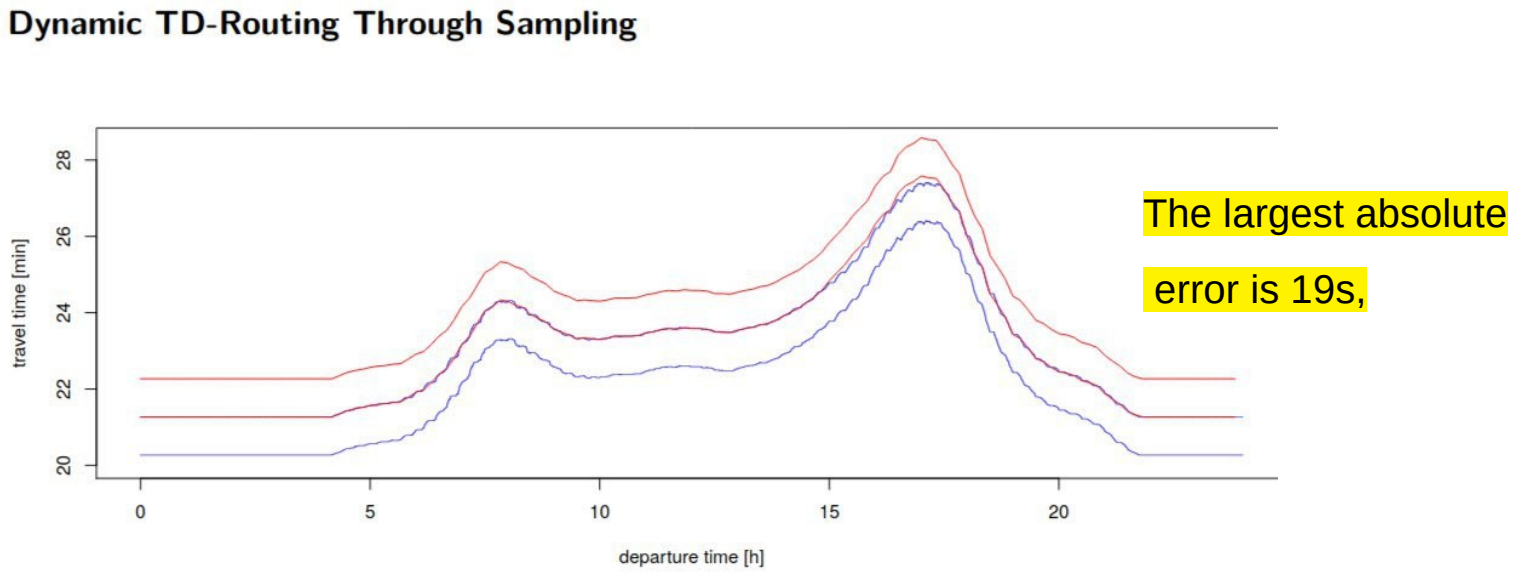
They use Dijkstra here !

# Computing profile: TD-S+P

The algorithm is fast because they only have to mark the edges once per query then run the time-dependent search numerous times.

For a sampling rate of 10min, the algorithm would mark the edges as for TD-S, then run the time-dependent extension of Dijkstra's algorithm restricted to the marked edges with the departure times 0:00, 0:10, 0:20...23:50

# TD-S+P




The largest absolute error is 19s,

Figure 2

Example profile over 24h. The red curve (top) was computed with TD-S+P, while the blue one (bottom) is the exact solution. The middle overlapping curves are the actual profiles. To improve readability

# TD-S+D

However, in many applications we must also take real time congestion into account.

Adapting TD-S by **modifying the computation of the sub graph H** yielding TD-S+D.

# TD-S+D

The sub graph H is the union of k paths

TD-S+D adds a shortest st-path according to the current real time traffic as k+ 1-th path to the union.

A efficient solution to the routing problem with real time congestion is needed.

# TD-S+D

In the pre processing step, TD-S+D computes one CCH in addition to the k CHs of TD-S.

At regular time intervals, such as every 10s, TD-S+D updates the CCH edge weights to reflect the real time traffic situation

# Simulating Traffic

Unfortunately, they do not have access to good measured real time traffic. they simulate real time congestion to study the performance of TD-S+D

# Simulating Traffic

- For an earliest arrival time query from s to t with departure time τ, they **first compute the shortest time-dependent path P** with respect to the historic travel times
- On path P they **generate three traffic congestion** by picking three random start edges.
- **From each of these edges they follow P for 4min**, yielding three sub paths

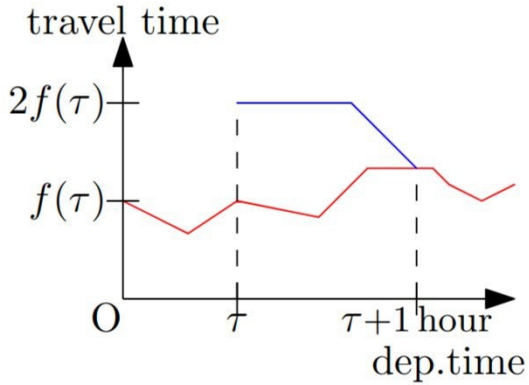
# Simulating Traffic

1- Travel time function f of e

2- by doubling the travel time at f(τ)and assume that it remains constant for some time

3- The congestion should be gone at τ+ 1h.

4- FIFO-property:e modified function must have slope of -1 before τ+ 1before joining the predicted travel time



The number of exactly solved queries decreases with instance size as the paths lengths grow with size

as most queries are answered exactly with TD-S

TD-S+4 has larger errors as it uses fewer time windows.

**Table 2** Number of exact time-dependent queries and absolute and relative errors for Freeflow, TD-S+4, and TD-S+9. "Q99" refers to the 99%-quantile and "Q99.9" the 99.9%-quantile.

| Graph | Algo | Exact [%] | Relative Error [%] | | | | Absolute Error [s] | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Avg | Q99 | Q99.9 | Max | Avg | Q99 | Q99.9 | Max |
| Lux | Freeflow | 80.0 | 0.244 | 5.1 | 11.5 | 28.1 | 5.0 | 106 | 235 | 356 |
| Lux | Avgflow | 81.0 | 0.123 | 2.5 | 6.4 | 19.4 | 2.5 | 49 | 143 | 329 |
| Lux | TD-S+4 | 97.7 | 0.008 | 0.2 | 1.5 | 4.9 | 0.2 | 4 | 30 | 141 |
| Lux | TD-S+9 | 99.6 | <0.001 | 0.0 | 0.1 | 1.7 | <0.1 | 0 | 3 | 27 |
| Ger | Freeflow | 67.9 | 0.085 | 1.5 | 3.1 | 12.4 | 11.1 | 200 | 417 | 825 |
| Ger | Avgflow | 69.2 | 0.044 | 0.8 | 1.9 | 10.3 | 5.9 | 113 | 284 | 587 |
| Ger | TD-S+4 | 94.6 | 0.005 | 0.1 | 1.0 | 3.0 | 0.8 | 17 | 159 | 474 |
| Ger | TD-S+9 | 98.2 | 0.001 | <0.1 | 0.4 | 3.0 | 0.3 | 1 | 76 | 374 |
| OGer | Freeflow | 60.7 | 0.140 | 2.0 | 4.7 | 12.4 | 15.9 | 219 | 465 | 1 104 |
| OGer | Avgflow | 68.8 | 0.050 | 0.9 | 2.2 | 6.5 | 5.7 | 96 | 227 | 619 |
| OGer | TD-S+4 | 96.4 | 0.002 | 0.1 | 0.4 | 2.0 | 0.3 | 6 | 47 | 333 |
| OGer | TD-S+9 | 98.5 | 0.001 | <0.1 | 0.2 | 2.0 | 0.1 | 1 | 24 | 276 |
| CEur | Freeflow | 54.9 | 0.089 | 1.4 | 2.7 | 10.8 | 26.4 | 428 | 833 | 1 477 |
| CEur | Avgflow | 55.8 | 0.048 | 0.8 | 1.7 | 6.6 | 14.2 | 235 | 507 | 1 069 |
| CEur | TD-S+4 | 91.1 | 0.006 | 0.2 | 0.7 | 3.8 | 1.8 | 47 | 226 | 547 |
| CEur | TD-S+9 | 96.8 | 0.001 | <0.1 | 0.3 | 1.2 | 0.5 | 6 | 109 | 397 |

# Analysis

TD-S with two selections of time windows:

TD-S+4 uses the windows 0:00–5:00, 6:00–9:00, 11:00–14:00, and 16:00–19:0

TD-S+9 uses the windows 0:00–4:00,5:50–6:10, 6:50–7:10, 7:50– 8:10,

10:00–12:00, 12:00–14:00, 16:00–17:00, 17:00–18:00

the query running times and the memory of TD-S+4 are lower.

TCH is lacking from memory .they couldn't run on 128 gig

TD-S+4 only needs about 2.4 times the memory required by the input.

TD-S+9 needs 4.1 times the memory.

TD-S+4 has lower pre processing time than TCH

**Table 3** Average preprocessing and running times and memory consumption of various algorithms.

| Graph | TD-Dijkstra | Freeflow | Avgflow | TD-S+4 | TD-S+9 | TCH |
|---|---|---|---|---|---|---|
| Average Query Running Time [ms] | | | | | | |
| Lux | 4 | 0.02 | 0.02 | 0.11 | 0.26 | 0.18 |
| Ger | 1 116 | 0.19 | 0.20 | 0.99 | 3.28 | 1.81 |
| OGer | 813 | 0.12 | 0.14 | 0.97 | 2.09 | 1.12 |
| CEur | 4 440 | 0.42 | 0.29 | 3.83 | 6.85 | OOM |
| Max. Query Memory [MiB] | | | | | | |
| Lux | 13 | 17 | 17 | 29 | 47 | 328 |
| Ger | 1 550 | 2 132 | 2 130 | 3 630 | 6 127 | 42 857 |
| OGer | 461 | 855 | 854 | 1 880 | 3 589 | 8 153 |
| CEur | 4 980 | 7 058 | 7 053 | 12 411 | 21 336 | >131 072 |
| Total Preprocessing Running Time [min] | | | | | | |
| Lux | — | <0.1 | <0.1 | <0.1 | 0.1 | 0.6 |
| Ger | — | 1.6 | 2.2 | 7.6 | 14.7 | 86.2 |
| OGer | — | 1.5 | 1.6 | 5.9 | 16.4 | 26.8 |
| CEur | — | 8.6 | 8.1 | 33.9 | 70.7 | 381.4 |

# Dynamic Time-Dependent Routing

In the dynamic scenario, we consider two types of congestion:

(a) the predicted congestion.

(b) the real time congestion.

The Predicted Path heuristic (Predict. P) as baseline, TD-S+D4, and TD- S+D9.

The Predicted Path heuristic computes a shortest path P with respect to only the predicted congestion NOT REAL TIME CONGESTION

P then is then evaluated with respect to both congestion types TD-S+D4, and TD-S+D9.

# Predicted Path P

Free flow and Predicted Path are similar in spirit.

Free flow solves the time-dependent routing problem by **ignoring predicted congestion**.

Similarly, Predicted Path solves the dynamic time-dependent routing problem by **ignoring real time congestion.**

The Free flow heuristic produces surprisingly small errors

On the Luxembourg instance only 1.6% of the queries are solved optimally.

TD-S+D reduces these errors.

The minimum number of optimally solved queries is 92.9%.

| Graph | Algo | Exact [%] | Relative Error [%] | | | | Absolute Error [s] | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Avg | Q99 | Q99.9 | Max | Avg | Q99 | Q99.9 | Max |
| Lux | Predict.P | 1.6 | 17.228 | 56.1 | 75.0 | 93.8 | 323.0 | 739 | 826 | 997 |
| Lux | TD-S+D4 | 94.7 | 0.017 | 0.5 | 2.3 | 6.2 | 0.6 | 15 | 93 | 231 |
| Lux | TD-S+D9 | 95.0 | 0.016 | 0.5 | 2.2 | 6.2 | 0.5 | 14 | 89 | 231 |
| Ger | Predict.P | 55.1 | 1.2 | 17.9 | 36.9 | 79.3 | 78.5 | 552 | 741 | 1 001 |
| Ger | TD-S+D4 | 90.9 | 0.032 | 1.0 | 2.6 | 7.0 | 3.5 | 116 | 233 | 474 |
| Ger | TD-S+D9 | 93.4 | 0.026 | 0.9 | 2.5 | 6.2 | 2.8 | 99 | 216 | 469 |
| OGer | Predict.P | 52.3 | 1.352 | 18.7 | 38.3 | 65.8 | 84.9 | 563 | 738 | 934 |
| OGer | TD-S+D4 | 91.5 | 0.031 | 1.0 | 2.6 | 5.4 | 3.2 | 108 | 224 | 462 |
| OGer | TD-S+D9 | 92.9 | 0.028 | 0.9 | 2.5 | 5.4 | 2.9 | 102 | 219 | 462 |
| CEur | Predict.P | 72.6 | 0.392 | 7.0 | 25.9 | 81.9 | 41.0 | 443 | 653 | 1 870 |
| CEur | TD-S+D4 | 89.5 | 0.015 | 0.5 | 1.6 | 5.2 | 3.3 | 106 | 244 | 547 |
| CEur | TD-S+D9 | 94.0 | 0.011 | 0.3 | 1.4 | 5.2 | 1.9 | 69 | 205 | 397 |

Number of exact dynamic, time-dependent queries and absolute and relative errors for the predicted path, TD-S+D4, and TD-S+D9.

**Table 7** Comparison of earliest arrival query algorithms. "n/r" = not reported. We report the running times as published in the corresponding papers (ori) and scaled by processor clock speed.

| | Numbers from | | Link & Merge? | Relative Error [%] | | | Run T. [ms] | |
|---|---|---|---|---|---|---|---|---|
| | | | | avg. | Q99.9 | max. | ori | scaled |
| TDCALT-K1.00 | [10] | OGer | • | 0 | 0 | 0 | 5.36 | 3.77 |
| TDCALT-K1.15 | [10] | OGer | • | 0.051 | n/r | 13.84 | 1.87 | 1.31 |
| eco SHARC | [8] | OGer | • | 0 | 0 | 0 | 25.06 | 19.7 |
| eco L-SHARC | [8] | OGer | • | 0 | 0 | 0 | 6.31 | 5.0 |
| heu SHARC | [8] | OGer | • | n/r | n/r | 0.61 | 0.69 | 0.54 |
| heu L-SHARC | [8] | OGer | • | n/r | n/r | 0.61 | 0.38 | 0.30 |
| TCH | Tab. 3 | OGer | • | 0 | 0 | 0 | 1.12 | |
| TDCRP (0.1) | [6] | OGer | • | 0.05 | n/r | 0.25 | 1.92 | |
| TDCRP (1.0) | [6] | OGer | • | 0.68 | n/r | 2.85 | 1.66 | |
| Freeflow | Tab. 2 & 3 | OGer | ○ | 0.140 | 4.7 | 12.4 | 0.12 | |
| Avgflow | Tab. 2 & 3 | OGer | ○ | 0.050 | 2.2 | 6.5 | 0.14 | |
| FLAT-$SR_{2000}$ | [22] | OGer | ○ | 1.444 | n/r | n/r$^3$ | 1.28 | 1.18 |
| CFLAT-BC3K+R1K,N=6 | [23] | OGer | ○ | 0.031 | n/r | 19.154 | 4.10 | 4.73 |
| TD-S+4 | Tab. 2 & 3 | OGer | ○ | 0.002 | 0.4 | 2.0 | 0.97 | |
| TD-S+9 | Tab. 2 & 3 | OGer | ○ | 0.001 | 0.2 | 2.0 | 2.09 | |

A*
CONTRACTION
ARC-FLAGS
LINK AND MERGE
+ Large memory

51 GB
16.1 GB
1.8 GB

# Conclusion

Introducing TD-S, a simple and efficient solution to the earliest arrival problem with predicted congestion on road graphs.

We extend it to TD-S+P which is the only algorithm to solve the profile variant in at most **50ms** on all test instances.

Further, we demonstrated with TD-S+D that additional real time congestion can easily be incorporated into TD-S.