Dynamic Time-Dependent Routing in Road Networks Through Sampling

Grimaldi Marco Alberto

DATA SCIENCE & SCIENTIFIC COMPUTING

Algorithmic Design

September 17, 2019

Outline

Outline

Abstract

"We study the earliest arrival and profile problems in road networks with time-dependent functions as arc weights and dynamic updates. We present and experimentally evaluate simple, sampling-based, heuristic algorithms."

Abstract

"Our experiments reveal, that the memory consumption of existing algorithms is prohibitive on large instances. Our approach does not suffer from this limitation. Further, our algorithm is the only competitor able to answer profile queries on all test instances below 50ms.."

Outline

Introduction

Routing in road networks is an important and well-researched topic and is formalized as a weighted directed graph; nodes correspond to position, edges represent road segments.

Figure: A road map represented as a graph

History

The most natural problem that arise in this topic is finding the shortest st-path form a node (s) to another (t). The first near-linear scaling algorithm able to solve this problem has been proposed by Dijkstra in 1959. The second breakthrough is the A* algorithm presented in 1968 by Hart, Nilsson and Raphael which is an extension of Dijkstra's using also an heuristic approach. Due to the huge growth in dimension of real networks even the finest implementation of A* can't compute the solution in acceptable time nowadays.

Contraction Hierarchies

A new-old approach: Contraction Hierarchies Order nodes by some metric and then iteratively remove the less important one creating shortcuts between his neighbours if necessary, i.e. if removed node was part of a shortest path between two of them.

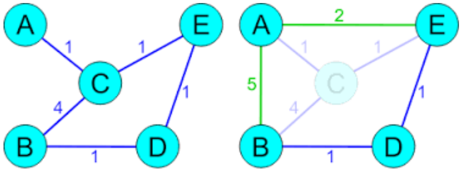


Figure: A road map represented as a graph

Now we have two step:

1. slow preprocessing phase, auxiliary data is computed
2. query phase, solved in sub-linear time thanks auxiliary data

Static-Dynamic scenario

Until now we suppose that edge values were *time-independent* but in real world enviroment is natural to take into account that those values can change during the day. All modern efforts are focused on solving the earliest arrival and the profile problem using predicted values and or real-time of a function that maps the entry time of a car into a node e onto the travel time $f_e(x)$.

Terminology

- *free-flow weight*: lower bound of $f_e(x)$;
- *dynamic time-dependent routing*: scenario of predicted and real-time congestion;
- TD: Time Dependent author's algorithm. -S: Sampled predicted congestion. +P: Profile problem. +D: Dynamic rel-time congestions.
- TCH: *Time-dependent Contraction Hierarchies*
- CCH: *Customized Contraction Hierarchies*.

Errors

"The routing problem with no congestion and with only realtime congestions can be solved efficiently and exactly. However, when taking predicted congestions into account, many proposed algorithms compute paths with an error."

But how much this error can be to being "acceptable"?
If a typical German traffic light needs to cycle though its program is between 30s and 120s we can say that errors in the order of a minute are perfectly fine for real world daily applications.

Complexity

"For the context of this paper, we use the following crude approximation: An algorithm is complex, if it combines functions using linking and merging operations 1 . Numeric stability issues are typically caused by these operations. Avoiding these operations thus also avoids these issues."

Based-on implementation

"Our program is build on top of the open-source CH and CCH implementations of RoutingKit. Fortunately, one can swap them out for any other algorithms that fulfills the requirements. An open-source TD-S implementation is available."

Outline

Free-flow and Avg-flow Heuristic

The free-flow travel time along an edge assumes that there is no congestion. Formally, the free-flow travel time of an edge e is the minimum value of e's travel time function f_e.

Free-flow heuristic works in two steps:

1. Find shortest time-independent path H with respect to the free-flow travel time.
2. Compute the time-dependent travel time along H for the given departure time.

Avg-flow

It is exactly the same as the Free-flow heuristic but uses the average travel time instead of the minimum travel time for every edge.

Time-Dependent-Sampling

The Free-flow heuristic never reroutes based on the current traffic situation. TD-S tries to alleviate this problem.

TD-S steps:

1. Compute a sub-graph H.
2. Run the time-dependent extension of Dijkstra's algorithm on the sub-graph.

Sub-graph is computed using a sampling approach; first the travel time function of every edge is divided in *k* time-interval (not necessarily equal), times are averaged obtaining *k* distinct time-independent graph, then shortest path is computed for every graph and finally their union is the sub-graph H.

Computing Profiles: TD-S+P

The query algorithm of TD-S+P also works in two steps:

1. Compute a sub-graph H.
2. Sample at regular intervals the travel time by running the time-dependent extension of Dijkstra's algorithm restricted to the sub-graph H.

TD-S+P is faster than iteratively running TD-S as the sub-graph is only computed once. Further, Dijkstra's algorithm iteratively runs on the same small sub-graph H.

TD-S+P Example

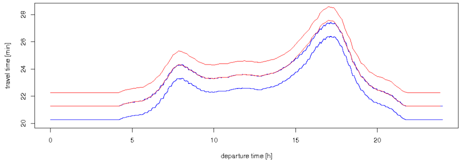


Figure: Example profile over 24h. The red curve (top) was computed with TD-S+P, while the blue one (bottom) is the exact solution. The middle overlapping curves are the actual profiles. To improve readability, we plot both curves a second time shifted by one unit on the y-axis.

Dynamic Traffic: TD-S+D

TD-S first step has to be adapted, the second is left unchanged:

1. Compute a sub-graph H as a union of k paths. TD-S+D adds a shortest st-path according to the current realtime traffic as $(k+1)^{-th}$ path to the union.
2. As TD-S:

In the preprocessing step, TD-S+D computes a 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 realtime traffic situation.

Simulated congestions and Environment data

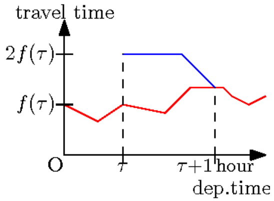


Figure: Travel-time weight function with simulated congestion.

Outline

Setup

Three production-grade instances have been used:

1. Lux: useful to investigate urban context
2. Ger: standard scale of searching queries
3. Central Europe: for test scaling behaviour

To compare related work they added also a forth one:

4. (decade) Old Germany

	Nodes [K]	Directed Edges [K]	TD-Edges [%]	avg. Break Points per TD-Edge
Lux	54	116	34	30.9
Ger	7 248	15 752	29	29.6
OGer	4 688	10 796	7	17.6
CEur	25 758	55 504	27	27.5

Figure: Node count, edge count, percentage of time dependent edges, and number of break points per time-dependent weight function for all instances.

Hardware and Queries

"Our experiments were run on a machine with a E5-2670 processor and 64GB of DDR3- 1600."

"For every instance, we generated 10^5 *queries with source stop, target stop and source time picked uniformly at random."*

TD-S Setups

They used two different time windows:

1. TD-S+4 uses the windows: 0:00–5:00, 6:00–9:00, 11:00–14:00, and 16:00–19:00.
2. 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, and 19:00–21:00.

The choice is made to reflect rush hours and they made them using manual trail-and-error search.

Outline

Results

Graph	Algo	Exact [%]	Relative Error [%] Avg	Q99	Q99.9	Max	Absolute Error [s] 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

Figure: Number of exact time-dependent queries and absolute and relative errors for Free-flow, TD-S+4, and TD-S+9. "Q99" refers to the 99%-quantile and "Q99.9" the 99.9%-quantile.

Runtime and Memory consumption

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

Figure: Average preprocessing and running times and memory consumption of various algorithms.

24h-profile run-time

	Freeflow		Avgflow		TD-S+P4		TD-S+P9	
Graph	SubG.	Total	SubG.	Total	SubG.	Total	SubG.	Total
Lux	<0.1	2.6	<0.1	2.6	0.1	3.0	0.3	3.4
Ger	0.2	18.1	0.2	18.3	0.7	19.5	1.7	22.2
OGer	0.1	10.0	0.1	9.5	0.8	11.2	1.8	12.4
CEur	0.4	36.8	0.4	36.8	2.1	49.9	5.3	53.4

Figure: Average 24h-profile running times in milliseconds. “SubG” is the sub-graph comp. time.

Optimal paths

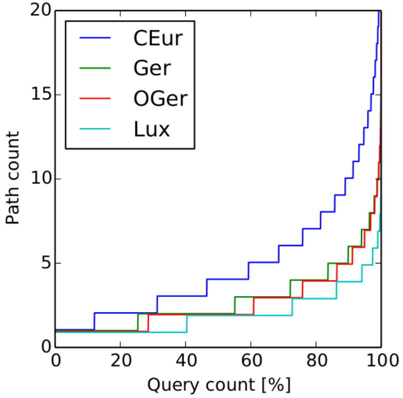
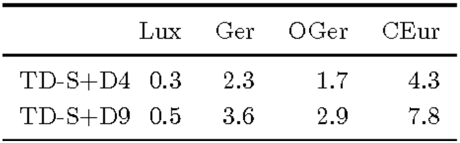


Figure: The number of optimal paths (y-axis) in function of number of queries (x-axis) for a 24h-profile of TD-S+P9.

Dynamic Time-Dependent Routing run-time



	Lux	Ger	OGer	CEur
TD-S+D4	0.3	2.3	1.7	4.3
TD-S+D9	0.5	3.6	2.9	7.8

Figure: Average query running times.

Dynamic Time-Dependent Routing errors

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

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

Comparison of TD-S with related work

	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	

Figure: 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.

Comparison of TD-S +P with related work

	Numbers from	Run T. [ms]	
		ori	scaled
eco SHARC	[8]	60 147	47 388
heu SHARC	[8]	1 075	847
ATCH ϵ=2.5%	[4]	38.57	30
TD-S+P4	Tab. 4	19.5	19.5
TD-S+P9	Tab. 4	22.2	22.2

Figure: Comparison of profile query algorithms. We report the running times as originally reported in the corresponding publications. Further, we report running times scaled by processor clock speed.

Outline

Conclusion

"We introduce TD-S, a simple and efficient solution to the earliest arrival problem with predicted congestions 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 realtime congestions can easily be incorporated into TD-S."

References I

Ben Strasser.
Dynamic Time-Dependent Routing in Road Networks Through Sampling.
In Gianlorenzo D'Angelo and Twan Dollevoet, editors, *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, volume 59 of *OpenAccess Series in Informatics (OASIcs)*, pages 3:1–3:17, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Dynamic Time-Dependent Routing in Road Networks Through Sampling

Grimaldi Marco Alberto

Algorithmic Design

September 17, 2019