# Estimating accurate traffic time by smart simulative route predictions

Nadav Voloch[1], Neev Penkar[1], and Guy Tordjman[2]

[1] Ruppin Academic Center, Emek Hefer 4025000, Israel
[2] The Open University, University Road 1, Ra'anana, 4353701, Israel

nadavv@ruppin.ac.il

**Abstract.** Accurate estimation of travel time is a crucial challenge in modern navigation systems, where real-time traffic conditions constantly change. Traditional shortest path algorithms, such as Dijkstra's algorithm, provide optimal solutions under static conditions but fail to account for dynamic congestion, leading to inaccurate estimated times of arrival (ETAs). In this study, we propose a Smart Real-Time Estimation method that incorporates predictive modeling of future traffic congestion to enhance ETA accuracy. Our approach utilizes a traffic simulation framework to evaluate the effects of congestion evolution on route optimization. We conducted simulations on different road networks under varying traffic conditions, comparing our method to the standard Dijkstra algorithm. The results indicate that in medium-traffic scenarios, our predictive model significantly improves ETA precision by anticipating congestion buildup. However, in highly congested environments, while our approach reduces systematic prediction errors, it introduces variability due to the complexity of traffic interactions. Additionally, the presence of traffic signals impacts both methods, with our approach demonstrating better adaptability when congestion patterns are predictable. Overall, our findings suggest that incorporating future traffic estimations into routing algorithms enhances the accuracy and realism of travel time predictions. This method provides a more dynamic and adaptive solution for navigation applications, particularly in environments where traffic conditions fluctuate frequently. By leveraging predictive modeling, our approach offers a viable improvement over conventional shortest path algorithms, making it a promising direction for optimizing real-time route planning.

**Keywords:** Fastest route, traffic control, navigation applications, smart traffic, transportation communication.

## 1 Introduction

Over the past decade, geographical navigation applications have become an integral part of our daily lives. While in the past they were primarily used for military or commercial transportation, today, almost every smartphone is equipped with built-in navigation systems, often through multiple applications.

The algorithms powering these applications have grown increasingly sophisticated. Whereas a decade ago they relied solely on pre-existing maps, today they incorporate real-time traffic data provided by users themselves, creating a dynamic representation of vehicles and road conditions. With this influx of real-time data, determining the fastest route between a starting point and a destination has become both more accurate and more complex. Route optimization is no longer just about finding the shortest path; it must also account for traffic congestion at various points along the way. Since traffic conditions are highly dynamic - constantly changing as vehicles move, these calculations must be updated continuously.

Applications like Waze leverage user-generated data to monitor traffic in real time, allowing them to compute routes that consider current road conditions when estimating travel times (ETA). This approach aims to provide users with the most efficient route. However, a fundamental challenge arises: the traffic conditions at the time of calculation may not reflect the situation when the vehicle approaches its destination. Additional vehicles converging from different starting points can increase congestion, leading to longer-than-expected travel times.

In this research, we address a fundamental issue in modern navigation systems: the inaccuracy of estimated travel times (ETAs) due to the dynamic nature of traffic. Traditional navigation applications (Waze, Google Maps, etc.) rely on real-time traffic data to compute routes. However, these methods often fail to account for how traffic conditions will evolve over time. As a result, ETAs frequently change during travel, leading to inefficient route planning.

We can see a simple example of the problem in Fig.1. At the upper part of the figure, we can see the status of routes at 07:30 AM, where Car 1 goes to the city via point B seems to have the shorter route since the road is clear, but there are a lot of cars at point A that plan to go on the same route. At the lower part of the figure, we can see the status of the routes ten minutes after, at 07:40 AM, where the cars from point A reach point B and create traffic congestion, enhancing the ETA by twenty minutes.

To tackle this problem, we proposed a novel approach ([1]) that determines the fastest route by incorporating future traffic estimations. Instead of basing calculations solely on the current state of the road, we predict the positions of vehicles at key points along the route at the expected arrival time. This forward-looking methodology allows us to generate more accurate ETAs and optimize navigation decisions.

In this model, the road network is represented as a graph, where vertices correspond to key points (e.g., junctions) on the roads, and edges represent road segments. To estimate the fastest route considering future congestion, we compute all possible paths from a source vertex to a target vertex and evaluate each based on both static distance and predicted traffic load.

For each path, we define the Total Edge Weight ($tew_i$) as the number of vehicles located at a distance from the target similar to the source's, reflecting potential congestion buildup near the destination. Additionally, we consider the classical shortest-path distance. The goal is to identify the path that minimizes the combined effect of physical distance and traffic congestion, weighted by average vehicle length, to approximate the true real-time fastest route.

The algorithm is as follows:

***Finding the fastest path with future traffic estimations*** **(Graph *G,* Edges *E*, Vertex *source*, Vertex *target*, Average length of Vehicle *Vehicle_Length*):**

1. Create a list of all paths *source* to *target*:
   1.1. $L^{path} = Dinic\ (G, source, target)$
2. For each path $L_i^{path}$ where $1 \le i \le |\ L_i^{path}\ |$ set attribute of total edge weight ($tew_i$):
   2.1. $tew_i = \sum_{j=0}^{|E|} w^E{}_j\ (L_i^{path})$
3. For each path $L_i^{path}$ find the distance $d_i$ between *source* to *target*:
   3.1. $d_i\ (L_i^{path}) = |L_i^{path}|$
4. $minETA\ \ Path = L_0^{path}$; $min\ \ realTime\ \ distance = d_0$
5. For each path $L_i^{path}$:
   5.1. if $(d_i + tew_i * Vehicle\ \ Length) < min\ \ realTime\ \ distance$
   5.2. $min\_realTime\_distance = (d_i + tew_i * Vehicle\_Length)$
   5.3. $minETA\_Path = L_i^{path}$
6. Return $min\ \ realTime\ \ distance$, $minETA\ \ Path$

As we can see in the algorithm, the *min_realTime_distance* is calculated with the addition of the planned traffic load of the vehicles that approach the target point. The algorithm that computed minimal travel time (as a derivative of the distance) from all the optional paths. We assume a uniform average vehicle length to model space occupancy and congestion more realistically.

To evaluate our model, we conducted simulations with various routes differing in length and anticipated traffic congestion. Our findings demonstrate that the shortest route does not always correspond to the fastest route when future traffic conditions are considered. In cases where a slightly longer path had lower predicted congestion, our algorithm provided a more accurate ETA and ultimately led to a shorter actual travel time.

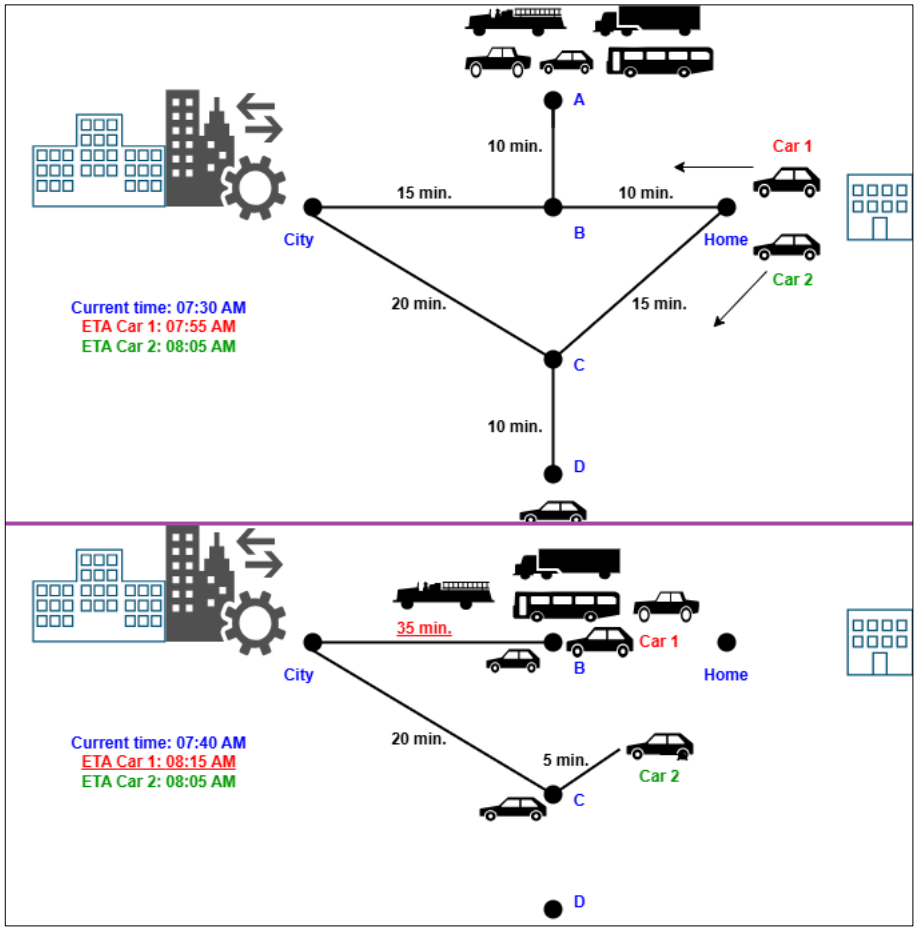


**Fig. 1.** The gap between planned ETA and actual ETA due to future traffic congestion not seen in the original route.

## 2 Background and related work

The shortest path problem involves determining a path between two vertices in a graph such that the total weight of the edges along the path is minimized, as summarized in [2]. One of the most well-known solutions to this problem is Dijkstra's algorithm, which was developed by Edsger W. Dijkstra to find the shortest path between nodes in a graph ([3]). Over the years, numerous research studies have optimized this algorithm, including those referenced in [4].

In our proposed approach, it is essential to consider all possible paths from a source point to a target point. Several efficient algorithms exist for identifying all potential paths between source and target nodes, such as the Ford-Fulkerson algorithm ([5]) and the Edmonds–Karp algorithm ([6]), which computes the maximum flow in a flow network with a time complexity of $O(VE^2)$, where V represents the number of points on the road and E denotes the number of different roads. Another relevant algorithm is Dinic's algorithm ([7]), which also addresses maximum flow networks but achieves a more optimized runtime of $O(V^2E)$. In our solution, finding all paths is necessary to determine the optimal route from a source to a target point, which aligns with the problem addressed by Dinic's algorithm.

A combined solution that utilizes Dinic's algorithm for pathfinding while incorporating weighted nodes to optimize for the Knapsack problem is discussed in [8]. This approach enhances the efficiency of path selection by factoring in node-specific weights.

Research on vehicle navigation systems has been explored in various studies, including early theoretical works such as [9], [10], and [11]. These early papers examined navigation from a conceptual standpoint, at a time when software and hardware technologies were still in their infancy and lacked the sophistication seen today.

In recent years, advancements in technology have led to significantly more accurate research in the field of automated navigation systems. For example, [12] explores the interaction between power systems and traffic networks, while [13] introduces a holistic system-wide load-balancing approach to optimize navigation. Our prior research ([14]) leveraged Large Distributed Dynamic (LDD) graphs to identify the fastest path within a dynamic traffic network. This work laid the foundation for our current research, as it incorporated time-dependent traffic states, providing a more accurate representation of real-world traffic fluctuations through a dynamic infrastructure.

Traffic load research has also gained momentum in recent years. A novel approach discussed in [15] introduces social priorities into smart junction traffic load estimations, ensuring that distributed traffic time allocations reflect principles of social fairness. This research direction was further developed in [16], where real-time parameters were incorporated to improve realism and accuracy of the algorithm. Additionally, in [17], a prototype simulator was constructed, demonstrating enhanced results in algorithm simulation. The algorithm was subsequently expanded in [18] to integrate auction-based decision-making for traffic light timing at intersections.

All the aforementioned studies serve as the foundation for our current research, which addresses inaccuracies in estimated travel times (ETAs) caused by inefficient navigation configurations. Our approach aims to enhance the accuracy and realism of ETA calculations, ultimately leading to more effective route planning and improved traffic load management. By refining the perception of navigation systems, we seek to provide users with the most efficient travel routes while mitigating congestion issues.

# 3 Methodology

## 3.1 Purpose

In this research we aimed to compare existing algorithms for configuring Estimated-Time-of-Arrival (ETA) with the implementation of our proposed method ([1]). As gathering real-time data in large quantities is not feasible, we chose to use a simulative approach. The basis of our work is simulating traffic in a "toy world", estimating the ETAs of chosen vehicles using several techniques, comparing each result to one another as also the actual time of arrival.

## 3.2 Simulation Software

The simulation library chosen was UXsim [19,20], that is based upon [21]. It is structured in Python [20], and this gives the ability to inspect the simulation at various timestamps. UXSim is an open-source macro/mesoscopic traffic simulator. It is based on Kinematic Wave model [22,23] (more specifically, mesoscopic version of Newell's simplified car-following model [24]) and dynamic user optimum-like route choice principle, which are well established methodology in the transportation research field (as seen in [19]). UXsim allowed us to insert nodes and links (with/without signals) with various needed attributes. Our simulation was based mostly upon link attributes such as jam density, free-flow-speed, average speed and number of vehicles on links, and node attributes such as signal timings [20].

Also, as our research is fundamentally based upon graph theory, we needed a network analysis library. We chose NetworkX [26] as it contained all the features we needed for the simulation and proof-of-concept.

## 3.3 Initializing the static attributes

We chose a relatively compact world consisting of 27 nodes (junctions) and 60 links (roads) out of which 4 nodes represented signals. The scenario was based upon the Or Akiva – Caesarea – Binyamina triangular area in Israel (henceforth called Area 1). The area was chosen mainly because of its rather "clean" gradient between crowded city roads and faster inter-city roads, thus simulating a fast highway, crowded city (residential) roads and the resulting intersections which are known to cause heavy and unexpected congestion. Also, speed limits were easier to obtain for the desired area. In later stages, the well-researched Sioux Falls network [27] (used in [28]) was also implemented. This network was chosen also because of its grid-like structure.
Initializing junctions in UXsim was straight-forward as the only needed attributes other than name were coordinates and signal timings if applicable. The coordinates were fixed using map data and were processed as described in sub-section Geometric Scaling.

UXsim offered several tweakable attributes for links between nodes. The main ones we initialized in our simulation were free-flow-speed and length, with links being connected to signal nodes having another attribute - their signal group. The length of a link was simplified to the physical distance between the nodes the link connected. The free-flow-speed of the link was set to the priorly obtained speed limit. The scenario necessitated three discrete speed limits, 90 km/h for inter-city roads, 50 km/h for residential roads and for more realistic results, some links with a speed limit of 30 km/h were also added. All speeds were scaled to the necessary units required by the simulation. All lengths were deemed accurate to the point described by Geometric Scaling.
The simulation included scaling the coordinates/coordinate system to simulate the real-life area. The coordinates after initial fixing were scaled according to the appropriate scale-bar/map-scale in map explorers, in horizontal and vertical directions.

### 3.4 Initializing the dynamic attributes

Random Source Destination Approach: Of all defined pairs of nodes (junctions), 20 were randomly chosen and kept aside. These pairs would define the demand when the simulation is executed.
Adding Time-Specific Demand: Declarable effort was put into making the demand as life-like as possible. The demand was planned to resemble the following distribution, which was made by simple post-processing on data taken from [25] (publicly available).
The graph in Fig.2 describes the number of vehicles travelling on some road, as a function of the time of day.

Samples were taken from the above mentioned distribution and a demand was added cumulatively from a chosen time up to an hour forward, where the source and destination were systematically chosen from the list described in the Random Source Destination Approach. The demand was added in small amounts, so that the accumulative structure of the simulation and the probabilistic structure of times chosen result in simulated demand which is fairly distributed over the world and random.

Pre-Defined Vehicles: For testing purposes, a small set of vehicles were added to the world, at random times throughout the simulation. The progress of these vehicles was monitored throughout the simulation and their ETAs were stored separately. These vehicles were added randomly over the duration of the simulation, with randomly chosen sources and destinations from a specific list of testing sources and destinations. These testing sources and destinations were chosen to be nodes from the outskirts of the world, to make the testing trip contain a variety of conditions. Also, some efforts were made to separate the testing sources from testing destinations (as far as possible).

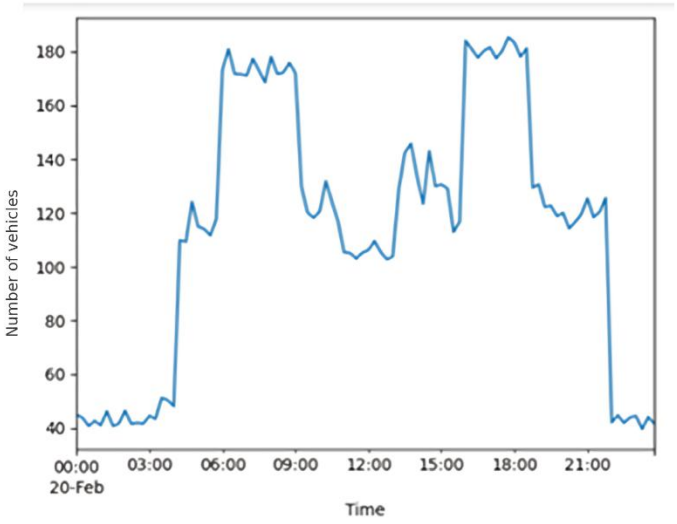


**Fig. 2.** The number of vehicles passing through a junction as a function of the time of the day.

### 3.5 Simulation flow

The simulation was executed on the above-mentioned structure with an hour resolution of 100 time units over a period of a day (2400 time units in total). The time step chosen was five time units such that every hour had 20 "simulation routines".
We analyzed the world being simulated at every iteration, calculating and storing the ETA of pre-defined vehicles for post-simulation analysis. Post-simulation, we analyzed the ETAs of these pre-defined vehicles against the actual time it took for them to complete their defined journey. We then analyzed the difference with several metrics.
Overall, we simulated two worlds with various scenarios, the results of which are in the Results section.

The simulation routine is the piece of logic, code and analysis which was programmed to run on every iteration of the simulation. Our routine consisted of two main parts: World Translation and ETA analysis. At the end of this stage, a table containing vehicle-wise ETA data was available for post-processing.

World Translation: This was the next stage, in which the simulative world was transformed into a graph structure for analysis, with the updated link parameters for every edge (as nodes had static values, they remained the same). This graph was then processed by the ETA analysis stage. The world was translated to a graph in the following manner: the junctions in the simulated world represented nodes in the graph.

**Table 1.** Parameter description of edges in the graph system

| Parameter | Description |
|---|---|
| *U* | Source node/junction |
| *V* | Destination node/junction |
| *Speed* | The average speed of cars travelling on the specific link |
| *Length* | Length of the specific link |
| *Density* | (Normalized) Density of traffic flow on the specific link |
| *Num_Vehicles* | Number of vehicles on the specific link |
| *Weight* | The expected time required to travel the link, length divided by average speed |

The links/roads were mapped to edges in the graph but with the following the parameters that are shown on Table 1. In addition to the stated before, the simulated world in UXsim [19,20] is translated to a graph in NetworkX [26] for further analysis.

ETA Analysis: this stage is based upon graph theory and is not strictly bound to traffic analysis (therefore the need for World Translation). The ETA analysis was based upon two algorithms: Vanilla Dijkstra [3, 29] analysis and the subject of this document [1]. Vanilla Dijkstra's algorithm is a classic shortest path algorithm that finds the minimum cost path from a single source node to all other nodes in a weighted graph with non-negative edge weights. It operates using a priority queue (typically a min-heap) to repeatedly expand the nearest unvisited node, updating the shortest known distances to its neighbors until all reachable nodes are processed.

## 4 Results

As mentioned in the previous section we compared our timings with the Vanilla Dijkstra algorithm and the results were as follows:

Vanilla Dijkstra: All (non-cyclic) paths between source and destination were found. The estimated time for travel for a specific path is the sum of the expected time of travel of the edges which make up that specific path. Therefore, the best path is the one which has the least time of travel. As stated in World Translation, the parameter "weight" represents the expected time of travel on a specific link therefore the sum of "weight" on edges of a path should give us the expected time of travel.

Smart ETA Estimation: As described in [1], this algorithm accounts for all cars travelling to the destination from the source (via various routes) and not only the ones which cause path-wise congestion on one single path. For simplicity, the effect of the secondary vehicles was modelled additively, i.e, the delay caused by additional vehicles as described in [1] was added to the base travel time, given by vanilla Dijkstra [3].

**Table 2.** Comparison between vanilla Dijkstra and our proposed algorithm

| **Scenario / Estimation** | **Vanilla Dijkstra** | | **Smart Real-Time Estimation** | |
|---|---|---|---|---|
| **Metric** | **MAE** | **ME** | **MAE** | **ME** |
| Sioux Falls Medium Traffic | 210 | 206 | 127 | 95 |
| Sioux Falls Heavy Traffic | 172 | 151 | 191 | 124 |
| Area 1 in Israel Medium Traffic No Signals | 359 | 358 | 389 | 309 |
| Area 1 in Israel Heavy Traffic No Signals | 787 | 787 | 839 | 825 |
| Area 1 in Israel Medium Traffic With Signals + Weight TF | 427 | 413 | 495 | 230 |
| Area 1 in Israel Heavy Traffic With Signals + Weight TF | 701 | 701 | 831 | 830 |

Various scenarios were based upon various worlds and evaluated in the simulation. We analyzed two main metrics – the mean error (ME) and the mean absolute error (MAE). All results are in uniform time units. The per-hour resolution of the simulation was set to 100 time units (2400 time units in total). Note that these metrics were chosen as the proposed algorithm is a predictive algorithm. Both algorithms are affected negatively by signals and heavy traffic. However, in medium traffic environments, our proposed algorithm is more precise.
The results are shown in Table 2. The simulation program and peripheral files are available on Git and can be found at [30].

The results in Table 2 demonstrate the performance differences between the Vanilla Dijkstra algorithm and our Smart Real-Time Estimation method across different traffic conditions and locations. In medium-traffic conditions for the Sioux Falls network, our proposed method shows a significant reduction in MAE (127 vs. 210) and ME (95 vs. 206), highlighting improved accuracy in estimated travel times. However, in heavy traffic conditions within the same network, our method results in a slightly higher MAE (191 vs. 172) but still achieves a lower ME (124 vs. 151), indicating better correction of systematic bias in predictions.

For Area 1 in Israel, the results follow a similar trend. In medium traffic without signals, Vanilla Dijkstra produced an MAE of 359, while our approach resulted in 389, reflecting a slightly higher variance due to dynamic congestion modeling. The ME also increased from 358 to 309, suggesting an adjustment in predicted travel times.

In heavy traffic without signals, both methods exhibit higher errors, with our approach yielding MAE and ME values of 839 and 825, respectively, compared to 787 for both metrics in Vanilla Dijkstra. When signals and weight adjustments were introduced, the proposed method performed better in reducing systematic error in medium traffic (ME of 230 vs. 413), although the MAE was still higher (495 vs. 427). In heavy traffic with signals, the results are comparable, with both methods showing similar ME values (830 vs. 701), but our approach leads to higher MAE (831 vs. 701), likely due to the compounding effects of modeled congestion.

These results suggest that while our Smart Real-Time Estimation method provides a more dynamic and congestion-aware estimation, it introduces higher variance in some cases, particularly under extreme traffic conditions. However, the lower ME values in several scenarios indicate that it provides more realistic travel-time adjustments, making it a promising alternative for adaptive route planning.

The proposed Smart Real-Time Estimation algorithm introduces added computational complexity due to its need to evaluate all possible paths and dynamically adjust edge weights based on predicted traffic. While this remains tractable in small networks, scalability becomes a concern in large urban graphs with thousands of nodes. To mitigate this, future work should explore optimizations such as limiting the analysis to the K-shortest paths, using heuristic-based pruning (e.g., A*), applying incremental graph updates instead of full reconstructions, and leveraging parallel processing to evaluate paths concurrently. These strategies would reduce runtime overhead and enable real-time deployment at city scale without sacrificing estimation accuracy.

## 5 Conclusion and future work

The results of our study highlight the limitations of traditional shortest path algorithms, such as Vanilla Dijkstra, when applied to real-world traffic scenarios. While Dijkstra's algorithm provides an optimal solution under static conditions, it does not account for the dynamic nature of traffic, where congestion evolves over time and significantly impacts estimated travel times (ETAs). Our Smart Real-Time Estimation approach introduces a predictive component that considers future congestion patterns, leading to more adaptive and realistic ETAs.

The comparison between both methods reveals several key insights. In medium-traffic environments, our method consistently produces more accurate ETAs by anticipating future congestion rather than relying solely on current road conditions. This allows for better route selection, particularly in cases where the shortest path may not necessarily be the fastest due to expected traffic buildup.

In heavy-traffic conditions, the benefits of predictive modeling are more nuanced. While our approach improves systematic travel time adjustments, the increased complexity of traffic interactions in highly congested scenarios introduces additional challenges. External variables, such as unexpected driver behavior and unaccounted bottlenecks, may contribute to higher variations in travel time estimations.

Traffic signals also play a significant role in route estimation. Both methods experience increased error when signalized intersections are introduced, but our approach demonstrates better adaptability in conditions where congestion follows predictable patterns. This suggests that while real-time prediction enhances travel time accuracy, further refinements are needed to handle irregular and extreme traffic conditions more effectively.

Our findings demonstrate that incorporating predictive traffic modeling into routing algorithms significantly enhances ETA accuracy, particularly in medium-traffic scenarios. Unlike traditional shortest-path methods that rely solely on static or real-time data, our Smart Real-Time Estimation approach anticipates future congestion by analyzing vehicle density and expected traffic evolution along alternative routes. This forward-looking strategy allows for more adaptive and realistic route planning.

In real-world applications, such a method could be integrated into modern navigation systems (e.g., Google Maps, Waze, or fleet management platforms) to improve commuter experience, logistics efficiency, and overall traffic flow. By proactively avoiding routes that are likely to become congested, navigation tools can reduce travel time variability, optimize fuel consumption, and lower urban emissions. Additionally, the approach could support smart city infrastructure by feeding more accurate forecasts into traffic light scheduling, congestion pricing models, or public transportation planning.

While further work is needed to integrate live traffic feeds and ensure computational scalability in large networks, this study lays the groundwork for more intelligent, congestion-aware navigation systems that respond not just to the present, but to what lies ahead on the road.

A key limitation of this study is its reliance on simulated rather than real-world traffic data, which may not fully capture the variability and unpredictability of live urban environments. Additionally, the proposed algorithm assumes uniform vehicle length and consistent driver behavior, simplifying real-world conditions where factors like accidents, weather, or non-standard vehicle patterns can significantly affect traffic flow. The absence of integration with live data sources (e.g., Waze or GPS feeds) limits the immediate applicability of the model. Finally, the method's computational demands may pose challenges in large-scale, real-time systems without further optimization.

The basic implementation uses a fixed-car length; however, in current and future work we aim to introduce traffic-related dynamics for further improvement of the results. Another work in progress is adding more parameters to the traffic and using dynamic graphs to implement our improved algorithms, and the implementation of the simulation is currently being modified to be fully operational Web- wise with optional client interactions that can control the parameter values.

**Data availability statement.** The datasets and software generated and analyzed by this research during the current study are available in [30].

**Acknowledgments.** The authors would like to thank Ruppin Academic center, Israel, and the Israeli Ministry of Innovation, Science and Technology (Proposal no. 0007846), for the continuing support in this research, implementation, and presentation. This study was funded by grant number 34836.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References


1. Voloch, N., & Voloch-Bloch, N. (2021, November). Finding the fastest navigation rout by real-time future traffic estimations. In *2021 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)* (pp. 13-16). IEEE. https://doi.org/10.1109/COMCAS52219.2021.9629051
2. Abraham, Ittai; Fiat, Amos; Goldberg, Andrew V.; Werneck, Renato F. (2010) "Highway Dimension, Shortest Paths, and Provably Efficient Algorithms". ACM-SIAM Symposium on Discrete Algorithms, pages 782-793. https://doi.org/10.1145/2985473
3. Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs". Numerische Mathematik 1: 269–271.doi:10.1007/BF01386390.
4. Mehlhorn, K., Sanders, P. (2008). "Algorithms and Data Structures: The Basic Toolbox" .199-200. Springer. https://doi.org/10.1007/978-3-540-77978-0
5. Ford, L. R.; Fulkerson, D. R. "Maximal flow through a network" . Canadian Journal of Mathematics. 8: 399–404. (1956). https://doi.org/10.4153/CJM-1956-045-5
6. Edmonds, Jack; Karp, Richard M. (1972). "Theoretical improvements in algorithmic efficiency for network flow problems". Journal of the ACM. Association for Computing Machinery. 19 (2): 248–264. https://doi.org/10.1145/321694.321699
7. Dinic, Y.,"Algorithm for solution of a problem of maximum flow in a network with power estimation".(1970). Doklady Akademii nauk SSSR. 11: 1277–1280. https://zbmath.org/?q=an:0219.90046
8. Voloch, N. (2017) Optimal paths of knapsack-set vertices on a weight-independent graph. WSEAS Transactions on Computers, 16, 163-171. https://www.wseas.org/multimedia/journals/computers/2017/a365905-071.pdf
9. Claussen, H. (1991). Vehicle navigation systems. In Modern Cartography Series (Vol. 1, pp. 225-235). Academic Press. https://doi.org/10.1007/978-3-642-12733-5
10. Dance, F. J., & Thoone, M. L. (1997). Vehicle Navigation Systems: Is America Ready? (No. 970169). SAE Technical Paper. https://doi.org/10.4271/861360
11. Burnett, G. E. (2000). Usable vehicle navigation systems: Are we there yet. Vehicle electronic systems, 3-1. https://trid.trb.org/View/709813
12. Shi, X., Xu, Y., Guo, Q., Sun, H., & Gu, W. (2020). A distributed EV navigation strategy considering the interaction between power system and traffic network. IEEE Transactions on Smart Grid, 11(4), 3545-3557. https://doi.org/10.1109/TSG.2020.2965568
13. Liu, Q., Kuboniwa, J., Kameda, S., Taira, A., Suematsu, N., Takagi, T., & Tsubouchi, K. (2015, December). Traffic navigation using positioning information and channel quality map for system-wide load balancing. In 2015 Asia-Pacific Microwave Conference (APMC) (Vol. 1, pp. 1-3). IEEE. https://doi.org/10.1109/APMC.2015.7411600
14. Voloch, N., Voloch-Bloch, N., & Zadok, Y. (2019). Managing large distributed dynamic graphs for smart city network applications. Applied Network Science, 4(1), 1-13. https://doi.org/10.1007/s41109-019-0224-2
15. Barzilai, O., Voloch, N., Hasgall, A., Steiner, O. L., & Ahituv, N. (2018). Traffic control in a smart intersection by an algorithm with social priorities. Contemporary Engineering Sciences, 11(31), 1499-1511. https://doi.org/10.12988/ces.2018.83126

16. Barzilai, O., Voloch, N., Hasgall, A., & Steiner, O. L. (2018, December). Real life applicative timing algorithm for a smart junction with social priorities and multiple parameters. In 2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE) (pp. 1-5). IEEE. https://doi.org/10.1109/ICSEE.2018.8646018
17. Fine, Z., Brayer, E., Proshtisky, I., Barzilai, O., Voloch, N., & Steiner, O. L. (2019, November). Handling traffic loads in a smart junction by social priorities. In 2019 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS) (pp. 1-5). IEEE. https://doi.org/10.1109/COMCAS44984.2019.8958182
18. Barzilai, O., Giloni, A., Voloch, N., & Steiner, O. L. (2020). Auction Based Algorithm for a Smart Junction with Social Priorities. Transport and Telecommunication, 21(2), 110-118. DOI 10.2478/ttj-2020-0008
19. Seo, T., (2023). UXsim: An open source macroscopic and mesoscopic traffic simulator in Python -- a technical overview, eprint 2309.17114, https://arxiv.org/abs/2309.17114.
20. Seo, T., (2023). UXsim Github Repository. https://github.com/toruseo/UXsim .
21. Seo, T., (2023). Macroscopic Traffic Flow Simulation: Fundamental Mathematical Theory and Python Implementation. Corona Publishing Co., Ltd., (in Japanese).
22. Lighthill, M. J., Whitham, G. B., (1955). On kinematic waves. II. a theory of traffic flow on long crowded roads. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences 229 (1178), 317–345. https://doi.org/10.1098/rspa.1955.0089
23. Richards, P. I., (1956). Shock waves on the highway. Operations Research 4 (1), 42–51. https://doi.org/10.1287/opre.4.1.42
24. Newell, G. F., (2002). A simplified car-following theory: a lower order model. Transportation Research Part B: Methodological 36 (3), 195–205. https://doi.org/10.1016/S0191-2615(00)00044-8
25. Aman, H. Traffic Prediction Dataset, https://www.kaggle.com/datasets/hasibullahaman/traffic-prediction-dataset/data
26. Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, “Exploring network structure, dynamics, and function using NetworkX”, in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008 http://conference.scipy.org/proceedings/SciPy2008/paper_2
27. LeBlanc, Louis J., et al. *An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem.* Transportation Research, vol. 9, no. 5, 1975, pp. 309–318, https://doi.org/10.1016/0041-1647(75)90030-0.
28. Hillel Bar-Gera, (2006) Primal Method for Determining the Most Likely Route Flows in Large Road Networks. Transportation Science 40(3):269-286. https://doi.org/10.1287/trsc.1050.0142
29. https://github.com/LdDl/ch/blob/master/vanilla_dijkstra.go
30. https://github.com/nadavvoloch/SmartTransportationSimulation