The Hitchhiker's Guide to the Optimal Route Planning

Oleksii Vedernikov, Lars Kulik and Kotagiri Ramamohanarao
Department of Computing and Information Systems
The University of Melbourne, Australia
Email: {vedernikovo, lkulik, kotagiri }@unimelb.edu.au

***Abstract*—Hitchhiking is the oldest ridesharing process without prior arrangements by the ride sharers. It usually involves uncertain waiting times and various combinations of lifts on roads. For this way of travelling, the problem of finding an optimal route is extremely important and has not been studied. We propose the concept of a *hitchhiking graph* to represent all possible decisions that a hitchhiker can consider on a road network. We develop an efficient pruning technique for a faster computation of the optimal route with the least expected journey time. The effectiveness of our methods is evaluated on road networks of selected countries.**

I. Introduction

Hitchhiking is the first well-known form of ridesharing which is an act of soliciting and getting rides without prior arrangement. The decision-making process for a hitchhiker involves a number of steps: choosing a pick-up location to thumb a lift, and after a driver stops and informs their route, deciding if they get in and where to get out, and repeat this process until they reach their destination. Assume that a hitchhiker has an initial and destination locations and a starting time, and they need to choose a path and lifts to reach the destination with minimum travel time. We define it as a hitchhiker's problem.

At first, a hitchhiker's problem resembles a straightforward shortest path problem, but closer inspection reveals there are many options for a hitchhiker: various combinations of lifts along a path, different quality of pick-up locations which affects waiting time, various paths that might be taken between a pair of locations. For example, for a path between two points, a hitchhiker could wait for a direct ride, try to take two lifts with a stop and seek for another ride, etc. The total number of such combinations of rides on a single path is 2^{n-2}, where n is a number of possible stop points. We also show that for each of these combinations, computing exact travel time is also exponential to the number of lifts.

A number of works on finding a shortest path for vehicles on road networks and optimal route planning in public transport systems has been studied [2]. However, they are conceptually different to our problem: a hitchhiker cannot choose a path, only lifts that follow certain paths, and faces an unknown number of lifts with unknown waiting times. Unlike public transport, waiting time for a hitchhiker does not follow a timetable, and is different for various points on the same road depending on how easy it is for a driver to stop. In addition, transport routes are fixed and their number is limited, while the number of hitchhiking routes is dependent on drivers' routes. Therefore, our problem is different from optimal route planning in road and transportation networks.

To the best of our knowledge, the idea of efficient route planning for hitchhikers that minimizes journey time has never been studied. We aim to find a robust algorithm to find an optimal route for hitchhikers, which might be used in a mobile hitchhiking recommender system. The main research question of this work is: given a road network and a start and destination points, how to efficiently find a route and a combination of lifts on it with the least expected travel time?

To answer this question, we first outline the whole hitchhiking process and describe the exponential complexity of the whole problem. Then, we introduce the concept of *hitchhiking graph*, which contains new data in addition to a road network, i.e. quality of pick-up locations, estimated traffic on roads, and constitutes a complete set of all possible rides - *hitchhikking edges*. We show that their number is cubic to the number of nodes in the worst-case.

Then, we develop a pruning technique to refine a set of hitchhiking edges, leaving paths leading to largest cities in the area and with adequate travel time within a spatial ellipse.

In summary, our key contributions of this paper are:

- We formalise the *hitchhiker's problem* and combine all properties of hitchhiking trips, including uncertain parameters: waiting time at pick-up locations, chance of being collected and number of lifts in a trip.
- We aim to integrate the most essential properties of a hitchhiker's problem into a proposed concept of a *hitchhiking graph*, which represents all possible decisions that can be made by a hitchhiker.
- We developed an efficient edge pruning technique to reduce the computational time via deleting the edges which are less likely to be used. The performance of our methods are studied on road networks of selected countries utilizing a wide range of parameters.

The remainder of the paper is organized as follows. Section II provides a review of existing research about hitchhiking and shortest path problems. Section III describes hitchhiker's problem and heuristics for its solution. Section IV has a summary of experiments on real road networks. Lastly, Section V describes future work for the given problem.

II. Literature review

A. Hitchhiking

Hitchhiking became popular in 1960s and attracted scientific interest from sociological community. There are several articles on investigating how different factors as gender, dressing, weather may affect likeliness of drivers to stop which are summarized by Kotz [11]. A recent paper by Vedernikov et al. [12] proposes a system that predicts a quality and waiting time at a pick-up location. However, the question of finding the optimal route for hitchhikers has not been studied yet.

B. Shortest path in road networks

Bast et al. [2] provides an excellent survey on current state-of-the-art approaches for route planning for road networks and public transport networks and do comparisons of these methods on various road networks. Historically Dijkstra's shortest path algorithm algorithm [7] is widely used, and current state-of-art methods include using of Hub Labels (HL) for fast lookup of distances from any node to nodes from a predefined set of hubs [6] or Transit Node Routing (TNR) which uses distance tables on a subset of the vertices [4]. Even though some methods may be applicable for hitchhiker's problem, shortest path routing on road networks is different due to the opportunity of choosing a road for a driver, which is not applicable for the hitchhiker's problem.

C. Optimal route in public transportation networks

Bast et al. [1] propose an idea of Transfer Patterns (TP): sequences of stations where change of vehicle occurs. TP are utilised together with direct connections without change of vehicle, and this method is used for public transport routing in Google Maps. Also, a new approach by Bast and Storandt [3] utilizes the use of expected frequency of transport instead of schedules. However, route planning in public transport networks is using predefined set of public transport routes, while a hitchhiker is operating on a much larger set of cars source-destination pairs. In additions, there are no predefined stops for hitchhiking, and waiting time on pick-up locations even in a few tens of meters may vary significantly due to difficulty of drivers to stop at those locations.

III. Problem description

A. Hitchhiking overview

First, we describe the typical process of hitchhiking. A hitchhiker starts their journey at an initial location and time and they want to get to a destination using lifts from random drivers on a road network. Next, the hitchhiker needs to decide which road they want to take and choose a first pick-up location to thumb for a ride. After a car stops, the hitchhiker asks a driver's route, and after that decides whether to accept the ride or wait for another car to stop. If the hitchhiker accepts the ride offer, they get in the car and continue their journey on the driver's original path. Since the driver will usually not change their route, but is able to stop at any point of their route, the hitchhiker has to decide where to get off as well. Usually their destinations are different, so the hitchhiker might need a few lifts to reach the final destination.

Next, we list *fundamental components* of hitchhiking:

1) Road network of a certain area;
2) Initial locations and destinations of hitchhikers;
3) Spatio-temporal trajectories of drivers' trips;
4) Willingness to offer and accept rides for pairs of drivers and hitchhikers;
5) Quality of all possible pick-up locations.

We postulate that the whole process of hitchhiking can be described using these principles. Since our goal is to develop foundations of a hitchhiking recommender system, knowing the complete information about all 5 components will lead to the precise solution. In the following subsections we will show the complex nature of the problem and make certain assumptions and restrictions.

B. Lift combinations on a path

Suppose there is a path between two nodes which has n nodes. The hitchhiker can wait for a direct lift to the destination or accept offers to intermediate nodes and ask for another lift from those. Since a hitchhiker may get off at any node, the total number of combinations of lifts is equal to the number of permutations with repetitions 2^{n-2}. For example, consider a road network segment in Figure 1 and the corresponding tree of all combinations of lifts in Figure 2:

Fig. 1: Example of a road network

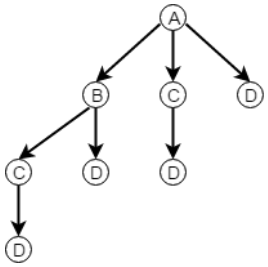


Fig. 2: Lift combinations

Source A and destination D correspond to the root and leaves respectively. Arrows from each node represent the direct rides, and number of paths from root A to leaves E is equal to the number of combination of lifts for a hitchhiker, which is $2^{(4-2)}) = 4$. Note that the tree has no loops, because at any situation, the total travel time of coming back and stopping at a certain location for the second time can not decrease the total travel time. Even if some of the combinations have the same intermediate stop point, the hitchhiker arrives there at different times, thus a following strategy is different.

C. Time computation for a hitchhiking trip

Next, we describe travel time for hitchhiking trip on a certain path. Assume for any connected cities i, j on a road network, a hitchhiker plans to hitchhike from i to j by a single ride. Thus, they need to start thumbing at point i and wait for a

driver going to j to stop at i. After waiting, the hitchhiker and driver will ride in the same car to j. Thus, total travel time for a hitchhiker consists of waiting time and riding time. Denote $w_{ij}(t)$ as waiting time at node i for a lift to j at time t and riding time $r_{ij}(t)$ between nodes i and j at time t. Suppose both $w_{ij}(t)$ and $r_{ij}(t)$ are discrete random variables with probability mass functions (PMF) $f_{w_{ij}}(x,t) = Pr[w_{ij}(t) = x]$ and $f_{r_{ij}}(x,t) = Pr[r_{ij}(t) = x]$ respectively, which are estimated empirically. For example, waiting time at node i for a lift to j for 10 hitchhikers at noon was following: 5 min. for 3, 10 min. for 5, and 15 min. for 2, then PMF $w_{ij}(5, 12pm) = 0.3$, $w_{ij}(10, 12pm) = 0.5$, $w_{ij}(15, 12pm) = 0.2$.

The general problem is to find a probability mass function (PMF) of hitchhiking travel time from v_{start} to v_{end} starting at time t, which is denoted as $f_{v_{start} \to v_{end}}(x,t)$ and includes all waiting times t_i and riding times τ_i. This PMF allows to answer all questions about a route: how long it will take on average, what is the probability of reaching the destination in certain time etc. We show if the total number of possible lifts is unknown, the problem of finding the total travel time in a general time-dependent hitchhiking setting becomes complex.

Consider an example with one road with two nodes and a hitchhiker is going from A to B. Assume that all drivers start and finish their trips only at graph nodes. Despite a hitchhiker starts their journey at time t, their ride starts not straight away, but after some waiting time at moment $t + t_1$. Surely, long waiting time may affect following travel time and traffic conditions may be different at that time. PMF of total travel time in this case is

$$\begin{aligned} f_{A \to B}(x,t) &= \sum_{t_1+\tau_1=x} Pr(w_{AB}(t) = t_1; r_{AB}(t+t_1) = \tau_1) \\ &= \sum_{\substack{t_1=0 \\ 0 \le y_1 < x \\ 0 < v_1 \le x \\ y_1 + v_1 = x}}^{y_1} \sum_{\tau_1=0}^{v_1} Pr(w_{AB}(t) = t_1; r_{AB}(t+t_1) = \tau_1) \end{aligned}$$

Each added intermediate points on a single path will add two more summations. For this example, if all random variables of waiting and riding time have no more than m values each, the total number of computations is $O(m^4)$, or in a more general case with k lifts it will be $O(m^{2k})$ assuming each lift accounts for both waiting and riding time. Therefore, estimating full PMF even for a small graph is intractable, and some constraints have to be relaxed.

In summary, we aim to find not only a path, i.e. roads that a hitchhiker needs to take, but also which lifts they should take. Because the hitchhiker can not choose the whole path, they may choose only a starting point at any of the roads adjacent to their initial location and a point of the driver's route where they may get off if they accept the offer. As we have shown, computation of travel time for a hitchhiker at a given combination of lifts is exponential to the number of stops on it, and the total number of different combination of lifts at a single hitchhiking path is exponential. Combining them, the worst-case complexity of calculating total PMF on a given path will be $O(2^{N^k})$. Considering the fact that there are multiple paths on a road network between two given nodes, our problem becomes intractable.

D. Constraints and simplifications

First, we analyse fundamental components hitchhiking outlined in Section III-A. Representation of Components 1 and 2 is straightforward regarding our research question. Road network should be represented as a directed graph, and we will consider a hitchhiker's problem for one hitchhiker with single initial location and single destination. Regarding Component 3, since the information about all rides is not available, in this paper we will use traffic estimation based on previous research in this area [10]. Our assumption is that drivers and hitchhikers start and finish their journeys and stop only at nodes of a given road network, and hitchhikers may ask drivers to drop them off at any node of original driver's path.

Component 4 is widely researched in sociology papers as outlined in Section II, while we assume that willingness of drivers to stop is independent from a hitchhiker and uniform among all drivers. Therefore, success of a hitchhiker depends only on pick-up location (Component 5). We assume that every pick-up location has its own relative quality score which is represented in a fraction of all drivers who stop at the location. While technically a hitchhiker can try to stop cars at any point of roads, we assume each road has been already assigned with the best pick-up location and corresponding quality score. Therefore, the actual waiting time for a lift depends on two values: the quality of a pick-up location and amount of traffic.

To reduce exponential complexity, we relax time-dependency constraint and assume that all waiting and riding times are time-invariant. Therefore, we can apply Bellman condition of optimality [5], which states that whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. In our case, it means that if we know all optimal routes from $v_1, v_2, \dots v_k$ to v_{dest}, the optimal route from v_{start} to v_{dest} is a minimum over all sums of $dist(v_{start}, v_i) + dist(v_i, v_{dest})$, and it allows to apply standard shortest path algorithms on a new graph. Note that Bellman condition is true only for intermediate stops on a path, not all points that a hitchhiker is passing through.

For a single path, a number of possible combinations of lifts will be reduced to $\frac{n(n-1)}{2}$. In the example from Figure 2 all tree nodes with the same name (C - 2 nodes, D - 4 nodes) will transform to single nodes. Therefore, there will be only 5 nodes and 10 edges, which is illustrated in Figure 3.

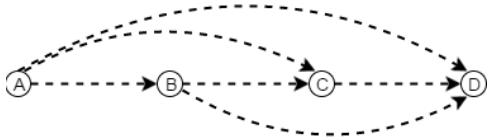


Fig. 3: Lift combinations

Here and after, dotted edges correspond to direct lifts between nodes. In addition, since travel times on intercity roads do not vary much, we assume that the travel time is constant depending on the distance and average vehicle speed. Since computing PMF on a combination of lifts is computationally intractable, we will compute the most important parameter: expected time of a trip. Therefore, one path and a corresponding combination of lifts is better than another if its expected travel time (sum of waiting and riding times) is smaller. Also, we assume there is only one hitchhiker seeking a ride, and all drivers who stop have free space to take the hitchhiker on board. In summary, in this paper we aim to combine the most essential properties of hitchhiking, namely:

- Hitchhiker can choose a next road to thumb up
- Hitchhiker can choose a desired next lift destination
- All rides start and stop only at nodes of a road network
- There are various combinations of lifts along a path
- Pick-up locations have different quality
- Each road has a predefined best location to hitchhike
- Waiting time depends only on a quality of a pick-up location and amount of traffic on that road
- Drivers' speed are uniform
- Drivers follow shortest paths on a road network
- Drivers willing to stop independently of a hitchhiker
- Objective function is expected time of a trip
- A driver who stops can take a hitchhiker on board
- Possibility to be dropped off before driver's destination

while leaving the rest for the future works, summarized in Section V. Our proposed model and developed heuristic algorithm are universal and can be successfully applied to all future works in the area of route planning for hitchhiking.

E. Formulation

Suppose we use a *road network* as a directed graph $G = (V, E)$, where $V = (v_1, v_2, ..., v_n)$ is a set of nodes (cities), and $E = (e_1, e_2, ..., e_m)$ is a set of edges (roads). An example is given in Figure 4.

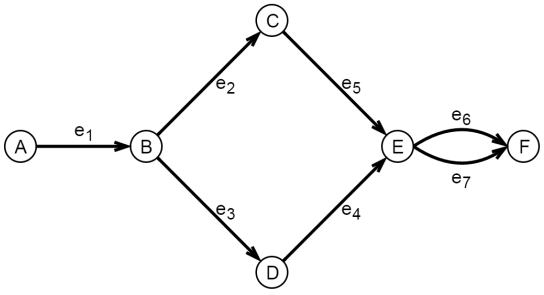
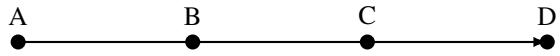


Fig. 4: Example of a road network

A **path** $P = \langle v_1, v_2, \ldots, v_{n+1} \rangle$ - a sequence of non-repeated nodes, where $i = 1\ to\ n,\ e_i = (v_i, v_{i+1}), e_i \in E$. Next, **hitchhiking route** is any subsequence of P which includes v_1, v_n as starting and ending note. For example, for a path $P = \langle ABCD \rangle$:



a set of routes would be $T = \{\langle AD \rangle, \langle ABD \rangle, \langle ACD \rangle, \langle ABCD \rangle\}$. We have shown that there are $2^{|P|-2}$ hitchhiking routes for each path.

Bellman conditions allow us to divide the original problems into smaller subproblems. Therefore, there should be multiple options for a hitchhiker to include all possible combinations of a pair of an adjacent node and a ride destination. Thus, we should add new hitchhiking edges for all such pairs and annotate them with expected travel time.

A **hitchhiking graph**: $HG = (V, E, W)$, where (V, E) is a connected directed weighted graph with a set of nodes V and a set of edges(roads) E, and $W = \{(v_i, v_j, e_k) | \forall v_i$ can reach $v_j; e_k$ is incident to $v_i\}$ is a set of *hitchhiking edges*, each one matching the lift from v_i to v_j via each of adjacent to v_i edges e_k. An example of HG for a road network from Figure 4 is given in Figure 5.

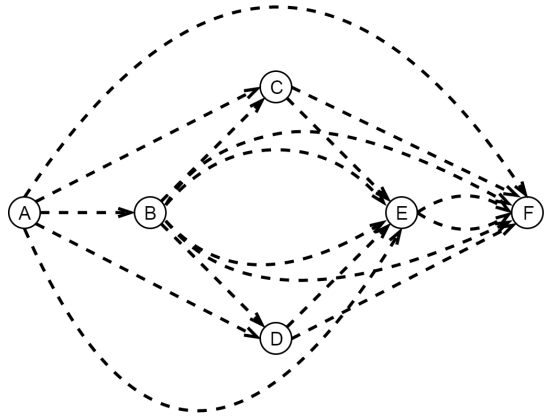


Fig. 5: Example of a hitchhiking graph

Note that there is only one edge from A to E and two from B to E which is equal to the number of adjacent edges from A and B respectively because the hitchhiker has to decide which road to take. Indeed, each driver will contribute to weight of $n-1$ hitchhiking edges if it passes n nodes of a road network on its way, since a hitchhiker can ask them to stop everywhere.

Since our assumption implies that the success of hitchhiking relies only on traffic and quality of a pick-up location, we need to introduce two more sets of input parameters for estimation weights w_i of hitchhiking edges. First, for each pick-up location at a certain road e_i, we denote q_i a quality score which is equal to the fraction of cars that will stop at the given location. Second, it is a number of cars per time unit $traffic(i, j)$ at a certain path. Since the amount of traffic between two nodes (cities) may be estimated with a gravity law via their populations and a distance between them [10], we will also include them as $Pop(v_i) \forall v_i$.

In this paper, we use only the most essential parameter of a hitchhiker's trip which is expected travel time. Therefore, we reformulated a problem of finding an optimal hitchhiking

route to a problem of finding a shortest path on a hitchhiking graph. We switch from a road network graph with edges V to a hitchhiking graph with hitchhiking edges W. While road networks have average degree of 3 for undirected networks, thus $|E| = 3|V|$ in the case of directed graphs. Hitchhiking graph becomes a complete multigraph taking into account initial graph's connectivity. Therefore, the number of hitchhiking edges in the worst case is $|W| = O(E \cdot V^2) = O(V^3)$, but in case of drivers taking the shortest path on a static road network there will be not more than one hitchhiking edge with a finite travel time for each pair of nodes. Complexity of Dijkstra algorithm using Fibonacci heaps is $O(E+V \log V)$, therefore for the hitchhiking graph it will be $O(V^3+V \log V) = O(V^3)$, which may be slow for large graphs. We propose an edge pruning method to handle this issue.

F. Edge pruning

Since there are many possible choices for a hitchhiker, our idea is to reduce the search space by pruning the hitchhiking edges which are unlikely to be visited. Intuitively, a hitchhiker will not probably go to a city which is very far from their source-destination (S-D) pair, or following the edge with long expected waiting time. Also, nodes with large population should be considered, since they have more drivers starting and ending their trips. To sum up, our idea is to reduce the search space by considering only the hitchhiking edges which are:

1) Inside of the start-destination ellipse, i.e. nodes v the $\forall(start, destination)\ pair\colon d(v, v_{start})+d(v, v_{dest}) \leq d(v_{source}, v_{dest})+\tau_\rho$, and τ_ρ is a possible detour budget.
2) Linking only to nodes with a population higher than a certain population threshold τ_{pop}
3) Going only via edges with travel time less than a certain time threshold τ_{time}

Note that (S-D) pair might be disconnected in a result graph, so query on a full graph might be required. In the Section IV, we conduct experiments with parameters $\tau_\rho, \tau_{pop}, \tau_{time}$ and discuss a trade-off between running time and accuracy of the results of the pruning method.

IV. Experiments

A. Experimental setup

We use road networks of European countries from DIVA-GIS [8] and Eurostat population grids [9]. We use the countries listed in the Table I.

TABLE I: Road networks and population grids used

Country name	No. of nodes	No. of edges	Total population	Total road length, km
Ireland	319	890	4,287,239	12682.47
Portugal	438	1194	9,123,842	16626.21
Netherlands	408	1138	15,807,422	8694.788

The road networks are represented as directed graphs. For each edge of the road network, assume q_i is a pick-up probability of the best pick-up location along it. Two settings are used: all pick-up probabilities are equal, or normally distributed $N(\mu, \sigma)$. In addition, to prevent potential disconnectivity, we assume there is a minimum possible pick-up probability which is equal to 0.001, i.e., in average one car out of 1000 cars will stop. Next, the population of each grid is assigned to the closest node of the road network. Then, the traffic between each pair of nodes in the road network is estimated using the gravity law [10]. Even though the authors do not mention the coefficients of the best fitting model, from their graph we conclude the estimated number of vehicles per year to be

$$traffic(i,j) = 0.00135 \cdot \frac{(P_i \cdot P_j)^{1.02}}{d_{ij}^2}$$

where P_i, P_j are populations of corresponding cities and d_{ij} is a distance in km between them. This traffic is assigned to all pairs of nodes on the shortest path.

After that, a full hitchhiking graph is created from a set of nodes from the road network. Each road's traffic is assigned to all hitchhiking edges which pass through it. The ravel time (i.e. weight) on hitchhiking edges consists of waiting time on the first edge of it and riding time. The waiting time is estimated using q_i and *traffic(i,j)*, and the riding time is estimated using the average driving speed (equal to $90km/h$) and length of the path corresponding to the edge.

At the second stage, we construct a pruned hitchhiking graph for each pair of source and destination points using threshold values. Since the absolute threshold values may vary depending on the dataset, we need to use relative values. Hence, for each dataset, 100000 shortest paths are computed for random S-D pairs on a full graph. Three values are extracted from each of the resulting paths: the population of all nodes on a path, the distance from each of the path nodes to the shortest path between the S-D shortest path on the road network, and the travel time. These values are saved in sorted arrays, and we extract p-th percentiles to obtain corresponding thresholds $\tau_{pop}, \tau_\rho, \tau_{time}$. Therefore, we are able to use the percentiles for each of the datasets rather than absolute values. We extract S-D pairs proportionally to the population in corresponding nodes (to verify the potential usage of a hitchhiking recommender system).

For the same S-D pair, we compare performance of the Dijkstra's shortest path algorithm [7] on both full and pruned graphs for the hitchhiking travel time HTT and the query running time QRT. Therefore, $HTT = \frac{HTT_f}{HTT_p} = 0.9$ means that travel time for a hitchhiker on a pruned graph is only 10% more than on a full graph, and $QRT = \frac{QRT_f}{QRT_p} = 5$ means a speed-up in 5 times for a query on a pruned graph.

For each of the sets of percentiles, we simulate 1000 S-D queries for each country. Since S-D pair might be disconnected on a pruned graph, we only use those sets of percentiles where number of such pairs is less than10%. After that, mean HTT and QRT are calculated. All experiments are performed on a PC with 3.4GHz CPU and 16GB RAM.

B. Results

In our model, q_i are distributed via $N(0.1, 0.1)$, and the results on different μ, σ are similar. We use percentiles of parameters as shown in Table II, where 0-th and 100-th percentiles

correspond to the minimum and maximum values from the experiments on a full graph, and other percentiles showed less effectiveness and thus are not included in this section.

TABLE II: Parameters settings

$P(\tau_{pop})$	[0,1,2,5,10,15]
$P(\tau_{time})$	[85,90,95,98,99,100]
$P(\tau_{\rho})$	[85,90,95,98,99,100]

The general distribution of QRT and HTT over the countries for different sets of percentiles is shown in Figure 6.

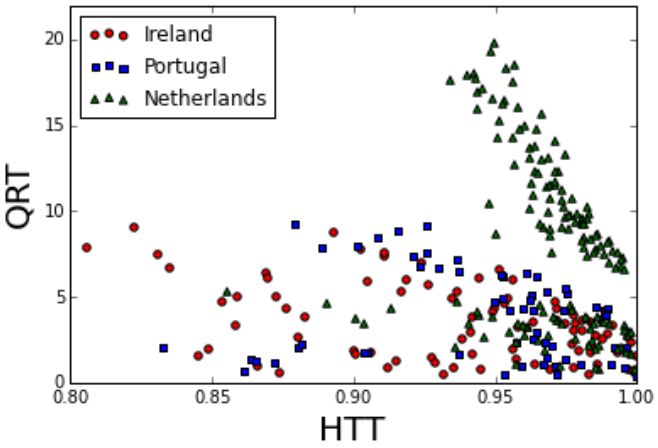


Fig. 6: QRT and HTT over sets of percentiles for countries

The points of best sets of percentiles are in top right corner. Generally, the performance of the pruning method is better in the Netherlands due to its higher traffic (distances are smaller, population is higher). Then, results over each sets of percentiles are averaged and shown in Figure 7.

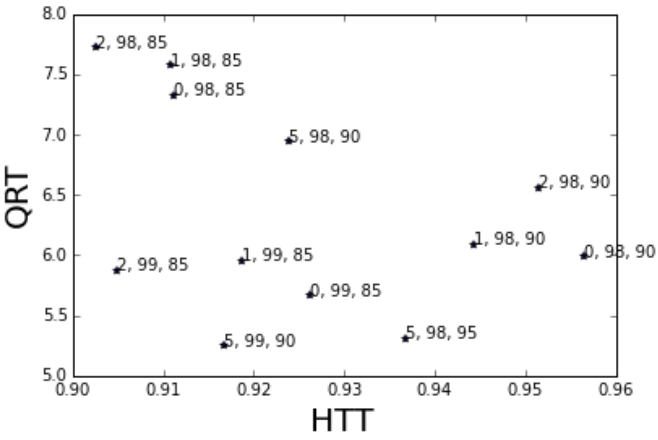


Fig. 7: Average QRT and HTT over sets of percentiles

There are a few non-dominated points, and we can take $(0.95, 6.56)$ which corresponds to the set of percentiles $P(\tau_{pop}) = 2, P(\tau_{time}) = 98, P(\tau_{\rho}) = 90$. Therefore, using a pruned graph, a hitchhiker can get a result of their query in 6.56 times faster, while the average travel time will be only $1 - 0.95 = 5\%$ slower than using a complete graph.

V. Conclusions and Future Directions

In this work, we provided an overview of hitchhiking, and formulated the hitchhiker's problem of finding an optimal route between two points on a road network. We designed a hitchhiking graph which integrates the most important properties of hitchhikers and provides a complete set of choices that could be made by a hitchhiker. While the total number of possible options for a hitchhiker is large, our pruning technique reduces the computational time while having similar accuracy. For example, using our pruning method,we reduce running time in 6.56 times comparing to the running time of a query on a full graph, while the average expected travel time for a hitchhiker is just 5% larger.

This work is the first one to analyse hitchhiking from a route planning perspective, and future works could include other heuristics to decrease the search space and speed-up the running query time. In addition, an important idea is to reduce precomputing time for constructing a full hitchhiking graph. Consequently, studying time-dependent graphs is an important direction. Next, a case with multiple possible pick-up location at all roads points might be considered. Finally, the impact of multiple hitchhikers using the same locations may be researched, as well as cases of multiple hitchhikers and limited capacity of cars.

References

[1] Hannah Bast, Erik Carlsson, Arno Eigenwillig, Robert Geisberger, Chris Harrelson, Veselin Raychev, and Fabien Viger. Fast Routing in Very Large Public Transportation Networks Using Transfer Patterns. In *Proceedings of the 18th Annual European Conference on Algorithms: Part I*, ESA'10, pages 290–301, Berlin, Heidelberg, 2010. Springer-Verlag. 00086.

[2] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Mller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. Route Planning in Transportation Networks. *arXiv:1504.05140 [cs]*, April 2015. 00117 arXiv: 1504.05140.

[3] Hannah Bast and Sabine Storandt. Frequency-based Search for Public Transit. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 13–22, New York, NY, USA, 2014. ACM. 00008.

[4] Holger Bast, Stefan Funke, Peter Sanders, and Dominik Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566–566, 2007.

[5] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[6] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 937–946, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.

[8] Diva-gis country data. http://www.diva-gis.org/gdata, 2017.

[9] Geostat 2011 grid dataset. http://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/population-distribution-demography/, 2017.

[10] Woo-Sung Jung, Fengzhong Wang, and H. Eugene Stanley. Gravity model in the Korean highway. *EPL (Europhysics Letters)*, 81(4):48005, February 2008. 00106 arXiv: 0710.1274.

[11] Fabian Kotz. The base-rate of hitch-hiking success and its moderators: A meta-analysis. *Transportation Research Part F: Traffic Psychology and Behaviour*, 46, Part A:149–160, April 2017. 00000.

[12] Oleksii Vedernikov, Lars Kulik, and Kotagiri Ramamohanarao. The Hitchhikers guide to the pick-up locations. *Open Geospatial Data, Software and Standards*, 1(1):12, December 2016. 00000.