Time-Optimal Route Planning for Non-Linear Recharging Electric Vehicles on Road Networks

Qinzhou Xiao[1], Yu Shao[1], Peng Cheng[1] (✉), Lei Chen[2], Wangze Ni[3], Wenjie Zhang[4], Heng Tao Shen[5], Xuemin Lin[6], and Liping Wang[1]

[1] East China Normal University, Shanghai, China
{qinzhou.xiao, yushao}@stu.ecnu.edu.cn {pcheng, lipingwang}@sei.ecnu.edu.cn
[2] HKUST & HKUST(GZ), Hong Kong SAR & Guangzhou, China
leichen@cse.ust.hk
[3] Zhejiang University, Hangzhou, China
niwangze@zju.edu.cn
[4] University of New South Wales, Sydney, Australia
wenjie.zhang@unsw.edu.au
[5] Tongji University & UESTC, Shanghai & Chengdu, China
shenhengtao@hotmail.com
[6] Shanghai Jiaotong University, Shanghai, China
xuemin.lin@gmail.com

Abstract. With the rise of green energy and technology, electric vehicles (EVs) are becoming a popular mode of transportation. In this paper, we investigate the route planning problem for EVs, considering factors such as driving range limits, charging station locations, and electricity levels. Unlike fuel vehicles, EV recharging exhibits significant non-linearity: it is faster at low electricity levels and slows at higher levels, requiring careful planning. To solve these challenges, we propose an efficient label assignment algorithm to construct a charging network, leveraging fast-charging capabilities. Additionally, an approximate method accelerates query processing. Experiments on real road networks show our algorithm achieves speeds 2–3 orders of magnitude faster than the baseline. [1]

Keywords: Electric vehicle · Route planning · Constrained shortest path.

1 Introduction

The demand and production of electric vehicles are rapidly increasing recently. Nowadays the green-house gas emissions have caused the rise of global average temperature, and the transportation component accounts for 23% of global green-house gas emissions. The electric vehicles are significantly more environmentally friendly compared to fossil-fuel-powered vehicles, i.e., the green-house gas emissions of electric vehicles is only one-fifth of that value of fossil-fuel-powered vehicles.

[1] Wangze Ni is also with The State Key Laboratory of Blockchain and Data Security; Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security.

Route planning is essential in many applications, such as minimal fuel consumption navigation, delivery service. Despite the importance, most of them only focus on one objective. For instance, while the shortest path might seem ideal, it may be congested, leading to increased travel time. Additionally, a driver might opt for a route with more right turns to minimize the risk of traffic accidents. However, due to the non-negligible charging times (e.g., 30 minutes even for fast charging) and limited cruising ranges (e.g., hundreds of miles), simply selecting the shortest path without considering the necessary charging time during the middle trip may lead to a long total time for electric vehicles' route planning.

Besides, it is essential to consider non-linear recharging, such as extreme fast charging (XFC), which adjusts electricity output based on factors like battery state, temperature, and charging protocol. XFC charges quickly at first but slows down as the battery fills to prevent overheating.

To efficiently solve the electric vehicle route planning with non-linear recharging, we propose a skyline-based exact algorithm to quickly find the time-optimal path from the source location to the target location. Specifically, we propose a search-based label assignment framework to find labels on each vertex. Additionally, we implement a bi-directional search strategy to accelerate the search process, complemented by an effective label concatenation method. Recognizing the high time complexity of the skyline-based exact algorithm, we further propose an approximate algorithm that significantly enhances the running speed, thereby optimizing the query response time for non-linear recharging electric vehicles. In summary, we make the following contributions in our paper:

- We formulate the non-linear recharging electric vehicle routing problem in Section 3.
- We construct the charging network using label assignment to find the time-optimal path between each vertex in Section 4.
- We compute the time-optimal paths for electric vehicle to enter and leave the charging network in Section 5.
- We conduct comprehensive experiments in real road network dataset in Section 6.

2 related work

The constrained shortest path problem (CSP) [9] refers to find the shortest path under the constraints of some certain variables. When the weight of an edge is no longer a single variable, the shortest path may be multiple solutions, each of which is called skyline path. The constrained shortest path problem can be divided into exact CSP and approximate CSP. [10] and [14] solve the exact CSP using dynamic programming. [4] and [15] convert the problem into integer linear programming and mixed integer programming problem. Sky-dij [5] maintains a heap to search paths originating from s, while a vertex can be visited many times. Another algorithm is based on *k-shortest path* (KSP) problem [8]. Pulse [13] uses DFS to search paths, the cKSP algorithm [2] is further optimized by changing the DFS to dijkstra's algorithm and introducing dominance relations. Some other work use index-based algorithms. Contraction hierarchy algorithm [15] is usually

used to compute index in shortest path problem. And the CSP-CH algorithm [3] extends CH algorithm to CSP. FHL [11] partition the graph into subgraph, do tree decomposition on each subgraph and assign label in the tree to construct index. Since the exact CSP has been proven to be NP-hard [6] [7], some existing work [17] [16]focuses approximate CSP. [5] proposes the first α-CSP algorithm with a complexity of $O(|E|^2 + \frac{|V|^2}{\alpha-1}\log\frac{|V|}{\alpha-1})$. [12] improves the complexity to $O(|V||E|(\log\log\frac{l_{max}}{l_{min}} + \frac{1}{\alpha-1}))$. COLA [17] is the first index-based algorithm to answer the approximate CSP problem.

3 PRELIMINARIES

Let $G = (V, E)$ denote a road network, where V is the set of vertices and E is the set of edges. Each edge $e_i \in E$ has a travel time $t(e_i)$ and electricity consumption $r(e_i)$. A path P from s to t is a sequence of edges $\{e_1, e_2, \ldots, e_k\}$, where $(v_k, v_{k+1}) = e_k \in E$. The total travel time and electricity consumption of P are $t(P) = \sum_{i=1}^{k} t(e_i)$ and $r(P) = \sum_{i=1}^{k} r(e_i)$, respectively. Electric vehicles recharge only at charging stations, $V_C \subseteq V$. Recharging is capped by the vehicle's capacity $\bar{c}$, and the initial electricity is c_0. The recharging time is represented by a function $J(c)$, denoting the time to charge from 0 to c. A route $R = (P, C)$ consists of a path P and charging amounts $C = (c_1, c_2, \ldots, c_k)$ at each vertex v_i. If $v_i \notin V_C$, then $c_i = 0$. The vehicle's electricity at v_i is $c_i^* = c_0 + \sum_{j=1}^{i-1}(c_j - r(e_j))$, with $c_1^* = c_0$.

In this problem, we need to find the shortest total travel time path, which is to minimize the driving time and recharging time:

$$T_R = \sum_{i=1}^{k} t(e_i) + \sum_{i=1}^{k}(J(c_i^* + c_i) - J(c_i^*)) \tag{1}$$

In addition, for a route to be correct, it must meet two conditions: 1) the electric vehicle must be able to reach the next vertex after recharging; 2) the recharging amount cannot exceed the upper limit:

$$r(e_i) \leq c_i^* + c_i \leq \bar{c} \tag{2}$$

4 Charging Network

4.1 Linear Charging Function

To obtain a basic solution, we impose the following constraints: all vertices are charging stations ($V_C = V$); the initial electricity is 0 ($c_0 = 0$); and the charging function is linear ($J(c) = \alpha \cdot c$).

In this case, equation 1 can be rewritten as:

$$T_R = \sum_{i=1}^{k} t(e_i) + \alpha \cdot \sum_{i=1}^{k} c_i = \sum_{i=1}^{k} t(e_i) + \alpha \cdot \sum_{i=1}^{k} r(e_i) = \sum_{i=1}^{k}[t(e_i) + \alpha \cdot r(e_i)] \tag{3}$$

$\sum_{i=1}^{k} c_i = \sum_{i=1}^{k} r(e_i)$ means to minimize the recharging time, the sum of recharging amount at each charging station should be equal to the total electric-

ity consumption without considering the energy loss. With equation 3, we just need to set the weight of each edge as the sum of $t(e_i)$ and $\alpha \cdot r(e_i)$, and then run the shortest path algorithm on this combined weights, then we can get the time-optimal path.

4.2 Non-linear Charging Function

We consider the case where the charging function is non-linear. In specific, we only consider the charging function to be concave upward, which is practical in real world situations.

Definition 1. *(concave upward function) $J(c)$ is a concave upward function if and only if its first-order derivative $J'(c)$ is monotonically increasing or its second-order derivative $J''(c) \geq 0$.*

If $J(c)$ is a concave upward function, it means that the lower the electricity, the less charging time it takes. A formal definition is as follows:

Definition 2. *(Low Electricity - High Efficiency) If $J(c)$ is a concave upward function, and $c_1 < c_2, \Delta c > 0$, then:*

$$J(c_1 + \Delta c) - J(c_1) < J(c_2 + \Delta c) - J(c_2) \tag{4}$$

Definition 2 is also in line with our intuition such as the charging process of smart phone. Therefore, if the electricity of vehicle is maintained in these fast-charging zones, then we can fully utilize the high efficiency feature to save a lot of time. We illustrate this in the following lemmas:

Lemma 1. *Given a route R, T_R can be smaller, provided the route is legal, by making the following adjustments:*

$$\begin{cases} c'_p = c_p - \Delta c \\ c'_q = c_q + \Delta c \end{cases} \quad (p < q, \Delta c > 0) \tag{5}$$

Therefore, by adjusting c_i constantly can reduce T_R. However, c_i also needs to satisfy the constraint $c_i^* + c_i \geq r(e_i)$ in equation 2. Thus, we can get:

Theorem 1. *Each time recharging, only need to recharge "just enough to get to the next charging vertex":*

$$c_i = r(e_i) \tag{6}$$

Another form of description is the electricity is zero each time reaching the next charging vertex. Thus we only need to change the weight of each edge e_i to $t(e_i) + J(r(e_i))$ and run the shortest path algorithm.

4.3 Sparse Charging Vertices

In the previous section, we give the algorithm defined on the full charging road network, i.e., $V_C = V$. However, in reality, the charging vertices are very sparse:

$$|V_C| << |V| \tag{7}$$

In this case, we need to construct a charging network, i.e., a new graph G' based on the original graph G, where the set of vertices in G' is $V' = V_C$ and any two vertex are connected to each other by a shortest path.

We first consider the case where the initial electricity is zero, then equation 3 can be written as:

$$T_R = \sum_{i=1}^{k} t(e_i) + \sum_{i=1}^{k} J(r(e_i)) = \sum_{i=1}^{k} [t(e_i) + J(r(e_i))] \tag{8}$$

This equation implies that the time consumed by each edge is independent, i.e., the actual traveling time $t(e_i)$ plus the recharging time $J(r(e_i))$. This also corresponds to the actual algorithm: each time we only need to recharge enough electricity to reach the next charging vertex. Then, the edge weights should satisfy:

$$\arg\min_{P_{ij}} \mathcal{W}(P_{ij}) = t(P_{ij}) + J(r(P_{ij})) \tag{9}$$

The next step is to find the smallest $\mathcal{W}(P_{ij})$ for each pair(i, j), i.e., find the skyline path set for any two charging station vertex (v_i, v_j) and convert the skyline path set into a time-optimal path using the charging function.

We assign a label $l = \{u, t, r\}$ to each intermediate solution, which denotes the path from S to u spending time t and consuming electricity r. Initial labels are $l_{init} = \{S, 0, 0\}$; for a label $l = \{u, t, r\}$, traveling an edge $e = \{u, v\}$ can be extended as $l' = \{v, t + t(e), r + r(e)\}$; the final result is all the labels like $l_T = \{T, t, r\}$. Finally, we compute $\mathcal{W}(l_T\{T, t, r\}) = t + J(r)$ and take its minimum value.

Bidirectional Search: Instead of using traditional unidirectional search, we use bidirectional search. Specifically, we expand the labels into $l = \{u, t, r, d\}$ where d represents the direction, and the final result is the couple of two labels:

$$\mathcal{W}(P^f \odot P^b) = \mathcal{W}(P\{t_f + t_b, r_f + r_b\}) = t_f + t_b + J(r_f + r_b) \tag{10}$$

Consider two paths P_t and P_r from S to T, which denote the paths with the least travel time and the least electricity consumption respectively, we have:

Lemma 2. *For all paths from S to T, the time lower bound is $\underline{t} = t(P_t)$, and the electricity lower bound is $\underline{r} = r(P_r)$; the time upper bound is $\overline{t} = t(P_r)$, and the electricity upper bound is $\overline{r} = r(P_t)$.*

Here S and T can be replaced with any other vertices, therefore the time/electricity upper bound at any intermediate vertex can be derived:

Theorem 2. *Among all the labels on vertex u, direction d, we can get time lower bound $\underline{t}^d(u)$ by searching the shortest path with time as the edge weight, at which time the corresponding electricity is the electricity upper bound $\overline{r}^d(u)$; the electricity lower bound and time upper bound are the same.*

Heuristic Search: To avoid the limitations of using just time t or electricity r as the weights, we introduce a heuristic function in our algorithm:

$$f(label\{u, t, r, d\}) = r + \underline{r}^{d'}(u) \tag{11}$$

r denotes the electricity consumed by the current path and $\underline{r}^{d'}(u)$ denotes the least amount of electricity needed to get from u to the target vertex. This heuristic function attempts to first expand the possible paths that consume the least amount of electricity.

Eliminate Dominated Solutions: Since the labels store the time and electricity of the entire path, they can be pruned in advance through the dominance relation. Specifically, if two labels $l_i = (u, t_i, r_i, d)$ and $l_j = (u, t_j, r_j, d)$ satisfies $t_i \leq t_j, r_i \leq r_j$ and l_i already exists, then we will skip l_j. Alternatively, we only need to expand the labels that are skyline paths, without expanding those that are dominated by other paths.

Monotonic Method: Assuming that there exists a label $l_1 = (u, t_1, r_1, d)$ and a new label $l_2 = (u, t_2, r_2, d)$ to be checked. Since l_1 is popped from the priority queue earlier than l_2, by the definition of the heuristic function, $r_1 + \underline{r}^{d'}(u) \leq r_2 + \underline{r}^{d'}(u)$, i.e., $r_1 \leq r_2$; thus, by the definition of domination, there is $t_1 \geq t_2$ (otherwise, l_2 is dominated by l_1). In other words, the time t_i of label l_i is necessarily monotonically decreasing. When expanding a new label l, the labels that are dominated can be terminated early by simply comparing t and the minimum time $t^d_{min}(u)$ of the existing label on direction d. The time complexity of this operation is $O(1)$, which can reduce the search space.

Label Coupling: Assuming that the expanding current label is $l = (u, t, r, d)$ and the label in another direction is $l' = (u, t', r', d')$. As mentioned in previous section, t is monotonically decreasing and r is monotonically increasing in vertex u and direction d. We can ignore some of the labels by the monotonicity.

The fact that r' is monotonically increasing means that $r+r'$ is monotonically increasing as well. Therefore, once $r+r'$ has exceeded the electricity limit $\overline{R}$, the electricity consumption for all subsequent couplings will also exceed the limit, allowing the loop to terminate immediately.

Label Pruning: When expanding labels, the labels can be determined in advance. If the label is unlikely to be an optimal solution, it can be directly ignored without being added to the priority queue like the following conditions indicates: 1.The minimum time or electricity to the target vertex has exceeded the upper bound, i.e., $t+\underline{t}^{d'}(u) > \overline{T}$, $r+\underline{r}^{d'}(u) > \overline{R}$; 2.The minimum total time to the target vertex has exceeded the current optimal solution, i.e., $\mathcal{W}(t+\underline{t}^{d'}(u), r+\underline{r}^{d'}(u)) > ans$; 3.The current time/electricity exceeds the current time/electricity upper bound, i.e., $t > \overline{t}^{d}(u)$, $r > \overline{r}^{d}(u)$; 4.The current t or $t + J(r + \underline{r}^{d'}(u))$ has exceeded the minimum value, i.e., $t > t^d_{min}(u)$ or $L(label) > L^d_{min}(u)$.

5 Outside the charging network

We proposed a navigation algorithm for the charging network and its construction in the previous section. However, since navigation tasks in real life can start and end anywhere with initial electricity, we also consider routes entering and leaving the charging network in addition to the main routes.

5.1 Non-zero Initial Electricity

When the initial electricity is not zero, i.e., $c_0 \neq 0$, the conclusion of the previous Lemma 1 is still correct, which means that it is still feasible to reduce the time

by continuously decreasing c_i. Note that, when calculating $c_k = r(e_k) - c_k^*$, since c_k^* may no longer be zero, it can lead to $c_k \neq r(e_k)$, or even $c_k < 0$. We extend this case to:

$$c_k = \max\{0, r(e_k) - c_k^*\} \tag{12}$$

c_i can still keep decreasing, but not less than zero. The whole route is thus divided into three stages:(1) Initial electricity stage. In this stage, the vehicle doesn't need to charge. This corresponds to the case where c_k gradually decreases to 0; (2) Initial electricity-recharging hybrid stage. In this stage (strictly speaking, only one vertex), the vehicle still has a certain amount of electricity, but not enough to reach the next charging station, so it needs to charge a portion of the electricity. This corresponds to the case where $c_k = r(e_k) - c_k^*$ but $c_k^* \neq 0$; (3) Recharging stage. This stage is the one we mentioned in section 4, where each time we recharge, we just need to make sure the electric vehicle can get to the next charging station. This corresponds to the case where $c_k = r(e_k)$.

If we need to charge a portion of electricity at v_i, we need to satisfy:

$$0 \leq c_i^* < r(e_i) \tag{13}$$

The bootstrap path leaving the charging network is simple, as the electricity after reaching the target vertex is no longer taken into account. It is only necessary to recharge enough electricity to reach the target vertex at the last charging station. However, entering the charging network from a non-charging station vertex $s = v_1 \notin V_C$ is complicated. We can't use the previous algorithm directly since the vehicle's initial electricity will have an impact on the results. Specifically, considering only the start vertex, target vertex and the charging station vertices while omitting the intermediate vertices, the path is $P = (P_0, P_1, ..., P_k) = (s = v_0, v_1, .., v_{k+1} = t)$. Here, P_i denotes the path from v_i to v_{i+1}, v_1 is the first charging station, and the initial electricity is c_0^*.

Note that in the third stage, each time when the vehicle reaches the charging station, its electricity is exactly zero, thus each recharging event is independent from each other, which means the global optimal solution can be achieved by obtaining the local optimal solution on each sub-path $P_i (i \geq 2)$. But since the electricity at v_1 is not necessarily zero, P_1 may not be an global optimal solution, thus we traverse all cases by online search.

5.2 Approximate Bootstrap Path

We propose an exact algorithm for bootstrap path generation, but since the online computation part remains highly complex due to the enumeration of v_2, an approximate algorithm is necessary to improve efficiency. One such technique is approximate label storage. The Pulse algorithm combines greedy and random strategies, storing up to Q labels per vertex. In two-dimensional problems, the first two labels are greedily selected to optimize the shortest time and least electricity, while the remaining labels are replaced randomly. This method limits the number of labels per vertex, reduces search complexity, and as shown in the experiment section, maintains high accuracy even with only three labels stored.

6 Experimental Study

6.1 Experimental Settings

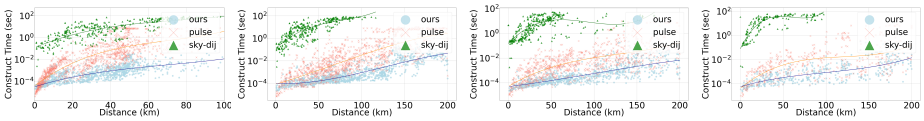
Datasets. We use 4 real road networks from the 9th DIMACS Implementation Challenge [1], corresponding to New York City (NY), San Francisco Bay Area (BAY), Colorado(COL), and Florida (FLA), with each graph having different numbers of vertices and edges. NY has approximately 260,000 vertices and 730,000 edges; BAY has around 320,000 vertices and 800,000 edges; COL has roughly 430,000 vertices and over 1 million edges; and FLA has about 1 million vertices and over 2.7 million edges. Each edge has attributes of travel time and distance. Electricity consumption is assumed proportional to mileage, ignoring conversion losses. In order to simulate the real situation, we also obtain the corresponding charging station distribution data and match it to the nearest vertex on the dataset. The coordinates of charging station and road network vertices are given in latitude and longitude terms, and we use *KD-Tree* to perform nearest vertex query.

Algorithms. All the algorithms are implemented in C++ with full optimizations. We compare our exact and approximate algorithm with Sky-dijk [5] and Pulse [13]

6.2 Construct Time

In this section, we analyze the construct time of the charging network. We randomly select 50 charging stations, limit the driving range to 200 km, and compute the skyline path set between each pair. Figure 1 (a)-(d) shows construct times for different algorithms as distance increases. For *Pulse* and *Sky-Dijk*, we cap iterations at 100 and 10 seconds, respectively.

Construct time grows with distance as more vertices are explored, enlarging the skyline set. *Sky-Dijk* is the slowest, as it explores numerous vertices without pruning labels, adding all to the heap. Our method is an order of magnitude faster than *Pulse*, which may enter "wrong zones" without precise answers. For short distances, *Pulse* is faster due to fewer operations and preprocessing.



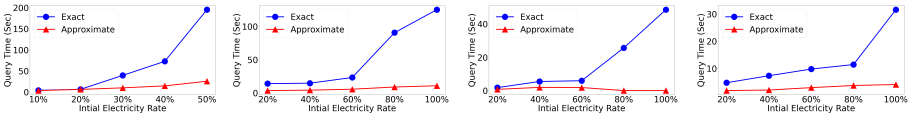
(a) NY (b) BAY (c) COL (d) FLA

Fig. 1: Construct Time

6.3 Query Time

In this section, we assume that neither the start nor target vertices are charging station vertex, thus we need to compute the bootstrap path online. Since once we have completed the construction of the charging network, the route planning within the charging network is a one objective shortest path problem. Therefore, we only compare our exact method and approximate method by randomly generating the start and target vertices.

Figure 2 (a)-(d) shows the average query time for different datasets with different initial electricity. It can be seen that as the initial electricity increases, the query time grows rapidly since the electric vehicle can reach farther charging



(a) NY (b) BAY (c) COL (d) FLA

Fig. 2: Query Time (Approximate vs Exact)

Table 1: Average Query Time

	Leaving Charging Network			Entering Charging Network			Overall	
Dataset	Approx	Exact	Speedup	Approx	Exact	Speedup	Error (10^{-4})	Speedup
NY	0.13	0.26	2.0x	5.74	77.74	13.5x	0.48	13.2x
BAY	0.24	0.61	2.4x	3.51	30.72	8.7x	1.41	8.3x
COL	0.16	0.50	3.0x	1.97	16.78	8.4x	2.89	8.0x
FLA	0.12	0.21	1.7x	1.14	10.85	9.4x	72.28	8.7x

stations with more initial electricity, making the query range larger. However, the approximate algorithm grows much slower than the exact algorithm, the reason is the approximate method only stores 3 labels in each vertex, making it fast to prune and couple labels.

Table 1 gives the specific query time, divided into two parts: entering the charging network and leaving the charging network. Since the leaving charging network process is simple and takes less time, the final overall speedup effect is mainly seen in the time it takes to enter the charging network process. We define the error as the difference between the total travel time of the navigation between the exact and approximate method. We can see that the error can be maintained in the order of 10^{-2} to 10^{-4} for speedup of about 10 to 20 times.

7 Conclusion

In this paper, we propose a route planning method for non-linear recharging electric vehicles. Our approach includes a label assignment framework, bidirectional heuristic search, and efficient dominance handling. We introduce optimization and pruning techniques to enhance algorithm performance and propose an approximate method for faster queries. Experiments show our method computes time optimal paths between charging stations in 0.01 seconds and completes full route planning in seconds, achieving a 100x speedup over existing methods.

8 Acknowledgment

Peng Cheng's work is partially supported by the National Natural Science Foundation of China under Grant No. 62102149. Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2023YFF0725100, National Science Foundation of China (NSFC) under Grant No. U22B2060, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, CRF Project C2004-21G, Guangdong Province Science and Technology Plan Project 2023A0505030011, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Zhujiang scholar program 2021JC02X170, Microsoft Research Asia Collaborative Research Grant, HKUST-Webank joint research lab and HKUST(GZ)-Chuanglin Graph Data Joint Lab.Xuemin Lin is supported by NSFC U2241211 and U20B2046.

References

1. 9th dimacs implementation challenge, http://www.diag.uniroma1.it//challenge9/download.shtml
2. Gao, J., Qiu, H., Jiang, X., Wang, T., Yang, D.: Fast top-k simple shortest paths discovery in graphs. In: Proceedings of the 19th ACM international conference on Information and knowledge management. pp. 509–518 (2010)
3. Geisberger, R., Sanders, P., Schultes, D., Delling, D.: Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In: Experimental Algorithms: 7th International Workshop, WEA 2008 Provincetown, MA, USA, May 30-June 1, 2008 Proceedings 7. pp. 319–333. Springer (2008)
4. Handler, G.Y., Zang, I.: A dual algorithm for the constrained shortest path problem. Networks **10**(4), 293–309 (1980)
5. Hansen, P.: Bicriterion path problems. In: Multiple Criteria Decision Making Theory and Application: Proceedings of the Third Conference Hagen/Königswinter, West Germany, August 20–24, 1979. pp. 109–127. Springer (1980)
6. Hartmanis, J.: Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). Siam Review **24**(1), 90 (1982)
7. Hassin, R.: Approximation schemes for the restricted shortest path problem. Mathematics of Operations research **17**(1), 36–42 (1992)
8. Hoffman, W., Pavley, R.: A method for the solution of the n th best path problem. Journal of the ACM (JACM) **6**(4), 506–514 (1959)
9. Irnich, S., Desaulniers, G.: Shortest path problems with resource constraints. In: Column generation, pp. 33–65. Springer (2005)
10. Joksch, H.: The shortest route problem with constraints. Journal of Mathematical Analysis and Applications **14**(2), 191–197 (1966). https://doi.org/https://doi.org/10.1016/0022-247X(66)90020-5, https://www.sciencedirect.com/science/article/pii/0022247X66900205
11. Liu, Z., Li, L., Zhang, M., Hua, W., Chao, P., Zhou, X.: Efficient constrained shortest path query answering with forest hop labeling. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE). pp. 1763–1774. IEEE (2021)
12. Lorenz, D.H., Raz, D.: A simple efficient approximation scheme for the restricted shortest path problem. Operations Research Letters **28**(5), 213–219 (2001)
13. Lozano, L., Medaglia, A.L.: On an exact method for the constrained shortest path problem. Computers & Operations Research **40**(1), 378–384 (2013)
14. Righini, G., Salani, M.: New dynamic programming algorithms for the resource constrained elementary shortest path problem. Networks: An International Journal **51**(3), 155–170 (2008)
15. Storandt, S.: Route planning for bicycles—exact constrained shortest paths made practical via contraction hierarchy. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 22, pp. 234–242 (2012)
16. Tsaggouris, G., Zaroliagis, C.: Multiobjective optimization: Improved fptas for shortest paths and non-linear objectives with applications. Theory of Computing Systems **45**(1), 162–186 (2009)
17. Wang, S., Xiao, X., Yang, Y., Lin, W.: Effective indexing for approximate constrained shortest path queries on large road networks. Proceedings of the VLDB Endowment **10**(2) (2016)