Analyzing the Vehicle Route Problem: A Heuristic Approach to Route Planning and Optimization

Anonymous Authors[1]

Abstract

We have chosen the subject area of route planning and optimization with a particular focus on the algorithm and model presented in the paper titled "Attention, Learn to Solve Routing Problems!" (Kool et al., 2019). The model in this paper builds upon existing published heuristic algorithms which address a variety of different route optimization problems, and presents a single solution that is broadly applicable. Our implementation attempts to take this model and further optimize it for specifically applications to the Vehicle Routing Problem (VRP). This subject area is becoming increasingly relevant in the field of artificial intelligence as more and more drivers and driver-less vehicles are relying on GPS calculated routes to get from point A to point B. We referenced six other papers which all focused on several key aspects of this subject area. The original three papers included topics such as pricing of routes dependent on traffic patterns[1], calculating optimal route lengths and times[2], and the process of continuously checking for new routes in order to potentially shorten travel time[3]. To aid in implementation and background research to gain a higher contextual understanding, we incorporated three additional papers in our research which are all centered around the topic of optimization of route planning, and either implement their own end-to-end framework[4] for solving the problem, or incorporating/optimizing existing algorithms.[5][6]

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1](Chen et al., 2020b)
[2](Yang & Nikolova, 2016)
[3](Chen et al., 2020a)
[4](Nazari et al., 2018)
[5](Lu et al., 2020)
[6](Delarue et al., 2020)

1. Related Work: Summaries and Reviews

1.1. Attention, Learn to Solve Routing Problems! (Kool et al., 2019)

1.1.1. Summary:

The paper that we have chosen to implement for this project, titled "Attention, Learn to Solve Routing Problems!" was accepted at ICLR 2019 and was written by Wouter Kool, Herke van Hoof, and Max Welling of the University of Amsterdam. The paper presents a model that allows for practical implementation of heuristics to address combinatorial optimization problems. For context, a heuristic algorithm is one that is designed to arrive at a solution to a problem quickly and efficiently, using shortcuts that can potentially sacrifice accuracy or optimality for the sake of reducing the total runtime. The primary problem that the authors apply their algorithm to within the paper is the Traveling Salesman Problem (TSP), with discussions in the Appendix on applications to the Vehicle Routing Problem (VRP), the Orienteering Problem (OP), and a variant of the Prize Collecting TSP (PCTSP). The overarching goal for the project was to demonstrate that the model developed is flexible enough that it can be applied to multiple different larger-scale route optimization problems while using a single set of hyperparameters, as this aids in the progress towards solving problems that do not yet have more robust or optimized solution models. With this in mind, the algorithm presented in this paper, in some cases, will not outperform other published models that are tailored to one specific routing problem. This is because of the heuristic nature of the algorithm combined with the fact that it is meant to be widely applicable.

The algorithm in question implements an attention based encoder-decoder model where the initial input is a graph of nodes with the graph structure being dependent on the specific optimization problem being addressed (ex: TSP, VRP, etc.). The encoder computes 128-dimensional node embeddings for each node in the graph which house all relevant graphical information for each node, and for which input order is not a factor. The encoder implements the attention model by updating the embeddings using multiple attention layers each with two sublayers. The purpose of

these attention layers is to mimic cognitive attention and focus computing power to specific nodes. The solution is defined as a permutation of the input graph nodes outputted by the decoder one-by-one. The Attention Model can be adjusted to be applicable to the different optimization problems by changing the input, mask (which takes into account the graph structure), decoder context, and the objective function.

The model is trained to improve over itself using a greedy rollout baseline that is meant to estimate the difficulty of the problem instance. Here, a greedy algorithm is one that provide the best local solution while continuing to find the globally optimal solution. This incorporates the idea of the attention-based model mentioned previously. Based on the assumption that for a more difficult problem instance, the algorithm will have a higher cost, the authors state that "The difficulty of an instance can (on average) be estimated by the performance of an algorithm applied to it."[7]

For the application to the VRP, the authors implemented datasets from (Nazari et al., 2018) to compare to their Reinforcement Learning framework[8], and found "significantly better results"[9] with greedy decoding. The authors also found that their greedy method performs well with larger cases as well and resulted in solutions that are comparable to the state-of-the-art VRP algorithm LKH3 published in 2017 by Keld Helsgaun, in "An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems: Technical report."

1.1.2. Review:

This paper does an incredible job of describing how the model developed can be adjusted to be applied to multiple kinds of route optimization problem. There was a plethora of background information in the related work section and we were able to get a strong idea of how this algorithm built upon and diverged from other related published works. It appropriately explains the purpose for this and the resulting benefits for problems that previously did not have majorly efficient solutions. The authors also managed to explain the efficiency and results of their algorithm as applied to the different routing problems in a way that was simple to understand. There could have been more detail in the sections that describe the algorithm itself; the paper was lacking a high-level end-to-end picture of the flow of the Attention Model. For someone who has a higher level understanding of neural networks this would not have been problematic as there was ample detail, but for someone lacking that background context, Section 3 dove straight into a description of the Attention Model's structure including many complicated phrases that may be unfamiliar. The Appendix of this paper includes extended results that paint a better picture of the application to the VRP and other routing problems, as the meat of the paper focuses specifically on the TSP for the sake of brevity.

[7](Kool et al., 2019)
[8](Nazari et al., 2018)
[9](Kool et al., 2019)

1.2. Approximation Algorithms for Route Planning with Nonlinear Objectives (Yang & Nikolova, 2016)

1.2.1. Summary:

The paper titled "Approximation Algorithms for Route Planning with Nonlinear Objectives" published in March of 2016 was written by Ger Yang and Evdokia Nikolova of the University of Texas at Austin Electrical and Computer Engineering department. This paper describes a polynomial-time algorithm that is an attempt to tackle the problem of nonlinear optimization of route planning, meaning route planning when there are multiple constraints or goals that need to be met. An example of this provided in the paper is route planning to try and optimise both cost and travel time. In general, it is a much more complicated problem than a simple shortest path algorithm, and there are many other factors to be considered as well, such as reliability of the roads and how risky the route is. The authors specifically mention how their algorithm handles the aforementioned particular case regarding the cost-to-time ratio, and the case of completing a route before a particular deadline using non-simple paths (meaning paths where a single graph node can be visited more than once).

The authors first explain how the general objective function for the problem, along with the specific example of the cost-to-time ratio objective function, are NP-hard. The significance of this is that it forces the acceptance into the problem of non-simple paths, and with that comes the polynomial-time algorithm designed by the authors. "We design a fully polynomial approximation scheme under hop constraints, and show that the algorithm can extend to run in pseudo-polynomial time under an additional linear constraint... we are able to design a polynomial-time algorithm to approximate the optimal solution [to a non-simple path problem] arbitrarily well."[10]

The introduction of non-simple paths requires additional constraints to be put in place, as the problem is no longer well defined. If nodes can be visited an infinite number of times, it opens up the possibility of the path p generated being an infinite loop, and so the authors of the algorithm implemented a hop constraint γ where the paths generated for a given graph can have up to γ hops. The

[10](Yang & Nikolova, 2016)

number of paths generated at any given time is at most polynomial and bounded by a multiple of the number of vertices. The second constraint that can be implemented to make the problem well-defined once again is a linear constraint in which the best path p must satisfy a certain linear constraint that involves a budget b–in this way, the path essentially has to come in at or below some specified budget (for example, time or cost).

Overall, this algorithm successfully addresses and provides a functional solution to the widely known problem of route planning while optimizing for multiple different aspects of travel.

1.2.2. REVIEW:

This paper does a great job of contextualizing the problem so that even someone who is relatively unfamiliar with algorithmic mapping and route planning can understand the significance of the work that has been done. The authors of this paper also succeeded at referencing the relevant work that they had based their findings on, which helped give us more background information for the problem at hand. It does a great job of outlining the algorithm (both mathematically and with pseudo code) that the authors developed to solve the non-simple path problems and used real-life examples to back up their work. It also provides a mathematical induction proof to strengthen their claims that their algorithm is efficient.

What was lacking was a better high-level, step-by-step description of their algorithm. The paper discusses the process of using non-simple paths and adding specific hop constrains and linear constraints, but then goes straight into the meat of the algorithm. We also would like to have seen more information about the real-world examples this algorithm was tested on and how much of a difference there was in the route that this algorithm generated versus a typical simple path shortest route algorithm. We also would like to know if this algorithm can be used to optimize for a vastly different constraint such as for low environmental impact (i.e. most fuel efficient).

1.3. Pay Your Trip for Traffic Congestion: Dynamic Pricing in Traffic-Aware Road Networks (Chen et al., 2020b)

1.3.1. SUMMARY:

The paper titled "Pay Your Trip for Traffic Congestion: Dynamic Pricing in Traffic-Aware Road Networks" published in April of 2020 was written by Lisi Chen[1,2], Shuo Shang[1], Bin Yao[3], and Jing Li[2] of [1]UESTC in China, the [2]Inception Institute of Artificial Intelligence in the UAE, and the [3]Shanghai Jiao Tong University in China. The paper tackles the problem of traffic congestion due to rapid urbanization by analyzing the pricing based on route congestion for rideshare applications such as Uber and Lyft. The proposed solution outlined is called a *Dynamic Pricing Strategy*, or **DPS**. At a high level, the DPS works as such: "Assume that n travelers are planning their trips at the same time, and each trip may have k potential routes to connect the source and destination (e.g., the fastest route, the cheapest route, the greenest route, unobstructed routes). The trips are charged according to their "congestion contributions" to global urban traffic systems, and the DPS problem finds a matching between n travelers' trips and the potential travel routes to minimize the global traffic congestion."[11]

In practice, the DPS targets two specific sub-problems. The first is determining how much a given route contributes to traffic congestion, and the second is minimizing the *global traffic congestion factor* (i.e. the total amount of congestion in an area) over many trips and all of their potential routes at a given time. On top of this, there is an implemented route pricing algorithm that determines the price of a route based on the distance and amount of congestion. To address the first of the two sub-problems, each route is given a price based on two factors, namely the *congestion ratio* (ratio of the number of vehicles on a given road to the road vehicle capacity), and the *upgrade margin* (how many more vehicles it would take to bring the specified road to the next congestion level). The routes are priced based on these two criteria, and the cheapest one is the most optimal solution. To address the second sub-problem regarding the *global traffic congestion factor*, the authors have created an initial matching algorithm where the trips are all assigned their respective minimum priced route, and then a route swapping algorithm that determines whether it would be more effective with regards to minimizing congestion to swap the selected route with a different route for a given trip.

The authors performed an experimental study of the DPS on real road networks in New York City and Beijing, and reported the efficiency from a variety of perspectives including (but not limited to) the number of route candidates, the number of congestion levels, and the swap parameter. The study also compared multiple route matching algorithms. They found that their algorithm was highly effective and accurately solved the problem of congestion route pricing.

1.3.2. REVIEW:

This paper does a great job of explaining the problem and the solution in terms that are understandable to people of different education levels. All throughout the paper, the authors contextualize the problem and explain using both

[11] (Chen et al., 2020b)

pseudo code and English how the algorithm works. They go into a great amount of detail so that someone who reads it can fully understand why each of the design decisions for the algorithm were made, how exactly they approached the problem, and how the end result is determined. The pseudo code was helpful to visualize the algorithms that they describe, but it is their descriptions that make this paper worth one's while.

The authors take the time to explain to the reader the background research that has been done in this field in the past, and how previous pricing algorithms differ from their own (i.e. how their work is furthering the bounds of knowledge on the subject). They also go into detail regarding the experimental study that was done to analyze the effectiveness of the DPS, which helps the reader even further comprehend the meaningfulness of the work the authors have done.

1.4. Real-Time Route Search by Locations (Chen et al., 2020a)

1.4.1. Summary:

The paper titled "Real-Time Route Search by Locations" published in April of 2020 was written by Lisi Chen[1,2], Shuo Shang[1], and Tao Guo[3] of [1]UESTC in China, the [2]Inception Institute of Artificial Intelligence in the UAE, and [3]Google in Singapore. The paper addresses the algorithmic problem of Continuous Route-Search-by-Location (C-RSL), which is the process of continuously searching for better routes real-time on a given trip. This is an important feature in route planning because as the authors describe, road conditions can change real-time, and doing a single static route search in the beginning of a trip cannot take into consideration the constant change in traffic patterns. A limitation of attempting to implement C-RSL is the high computational complexity–many routes are being constantly calculated and recalculated and the time it takes to do this can get to be too long. To address this, the authors implemented a Query Location Expansion Matching (QLOE) algorithm that implements Dijkstra's network expansion work (Dijkstra 1959), and then compares the output of the QLOE to the routes found from the C-RSL. This cuts computational time because there are much fewer routes that need to be processed.[12]

The authors implemented a Direct Route Matching (DRM) algorithm to address the C-RSL problem, and performed a complexity analysis on the computations required for the DRM. This generates a set of queries, and the QLOE matches the queries to routes generated using network expansion (Dijkstra 1959) calculated by itself. The authors also implement a QLOE+ algorithm that has a more optimized expansion process. The algorithms developed were tested on real-world roadways in New York City and in Beijing, and the effectiveness of each of the DRM, QLOE, and QLOE+ were analysed and compared. After thorough experimentation, it was found that the QLOE and the QLOE+ algorithms were able to efficiently provide results for the C-RSL problem in a manner that is scalable and optimal.[13]

1.4.2. Review:

The authors of this paper do a good job addressing the problem of continuous route planning while keeping in the forefront of their mind the immediate computational time constraint. They clearly state the problem that they are addressing and provide a scalable and accurate solution to the problem. They also provide pseudo code and well written descriptions of the mathematical processes within the algorithms they present, so that the reader has a fundamental understanding of their solution to the C-RSL problem. They also do a great job of contextualizing the computational complexity of the DRM by naming and explaining all of the variables in the big-O notation. Moreover, they provide concrete evidence of the efficiency of their solution by testing it out on real-world roadways in two major cities.

The main improvement that could be made in this paper is the readability of the mathematical descriptions. The pseudo code helps immensely to visualize the algorithms and understand step-by-step what is being calculated and why, but the descriptions of the processes in the paper are much harder to follow. Another improvement would be to include the related work in this field in the beginning of the paper instead of at the end, as it would have helped us understand the problem more wholly if we had read first the related research and how the authors' work built off of it.

2. Implementation

2.1. Implementation Goal

Our goal for this project was to be able to implement and improve upon the algorithm presented in the paper "Attention, Learn to Solve Routing Problems!"(Kool et al., 2019). We chose this paper because as mentioned previously, it addresses route planning for multiple different route-based problems, has the most support with regards to previous code implementations, and the GitHub repository has been updated as recently as June 9th of 2021. We hoped to implement the routing algorithm that uses heuristics as presented in the paper to the VRP and generate optimal vehicle routes

[12](Chen et al., 2020a)

[13](Chen et al., 2020a)

for a demo map, but is scalable so that it could potentially be applied to a real street map. We chose this goal in order to further understand how route optimization is tackled so that we can be more qualified for relevant roles in the electric/autonomous vehicle industry. There are a growing number of software engineering positions that involve route planning, and so this project would be highly applicable to those and allow for greater contribution to that area in the workplace.

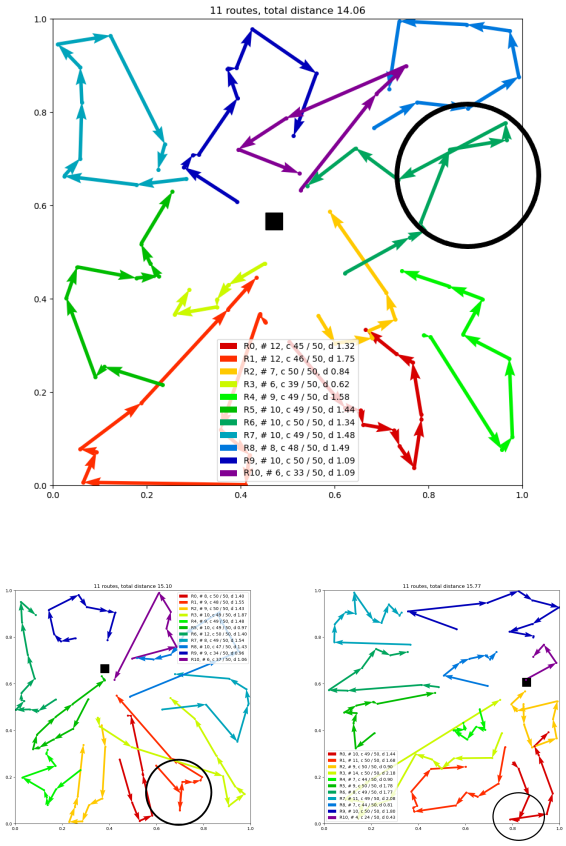
2.2. Implementation Plan

Our implementation plan had multiple steps. The first was to get the code running from the following GitHub repository (https://github.com/wouterkool/attention-learn-to-route), which is owned by the publishers of the paper. In order to do this, we needed to read through and understand the code by looking at the README, as well as scanning the code and reading through comments. We also spent some amount of time analysing other implementations of this algorithm published on GitHub by other programmers to understand how they interacted with the repository. From there, we needed to follow the steps outlined by the authors to install the necessary Python packages and libraries, as well as PyTorch, in order to run the code. Once the code was running, we would adjust the code and analyze the performance of the routes by comparing them to the results obtained by the authors.

More specifically, in the paper the authors mention that their results for the Capacitated VRP (CVRP) have room for improvement. "In some cases, we see that the order of stops within some individual routes is suboptimal, which means that the method will likely benefit from simple further optimizations on top, such as a beam search, a post-processing procedure based on local search (e.g. 2OPT) or solving the individual routes using a TSP solver."[14] Using this we were able to determine that the best approach to improve the algorithm for the VRP would be to implement one of the elements suggested by the authors. The authors since updated the algorithm to include an optional beam search which can be triggered by using:

```
--decode_strategy bs --width {beam_size}
```

This feature should allow for better results for CVRP applications. In order to get a baseline for the CVRP implementation, we utilized the example CVRP solution in the codebase by executing the `plot_vrp.ipynb` file which loaded a pretrained CVRP model with 100 nodes and plotted the results. We were able to observe the suboptimal paths that the authors mentioned in the paper, and they are visible in the below figures. The location of the flaw in the path is indicated by a black circle.



2.3. Implementation Issues

We ran into issues while attempting to simulate the experiments that were run and evaluated in the paper. We were able to copy the code from the original GitHub repository into a new private repository and clone that repository to our local machine without error, and we were able to install the necessary packages for startup with relative ease. The code base included instructions for getting started complete with specific commands to run in terminal, but the real issue came with the runtime for the code. When we initially attempted to run the code, the training time per dataset was over four hours. This was difficult to overcome because it was not until we left the code running overnight that we realized an extraordinarily long runtime was the issue–at first we assumed there was a logical error somewhere in the code causing it to loop infinitely.

The next significant roadblock in the progression of this project was related to the first, in that it took much longer than expected to train our improved model that would incorporate the beam search strategy. This made

[14] (Kool et al., 2019)

progress incredibly slow, as it would take many hours in order to train a model after even the slightest adjustments. From there it would add even more time to compare the new model with the original model, and when the results were undesirable (i.e. the new routes were worse than the originals) or were inconclusive (i.e. the new routes were identical to the originals), the cycle would repeat and we would have to adjust, train, and test the model all over again.

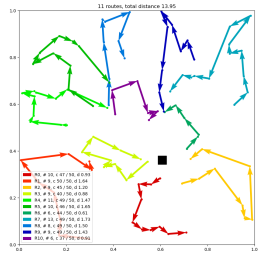
3. Evaluation

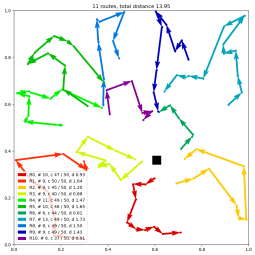
To evaluate that our implementation was successful, it was determined that we would use the same test data that was used in the `plot_vrp.ipynb` example of the Capacitated VRP with 100 nodes. As seen in the figures above, there were instances where the order in which the nodes are placed in the route was not optimal. This is because the CVRP model used in the example did not implement one of the suggested enhancements, namely beam search, 2OPT, or using a TSP solver for individual routes.

If our implementation could generate routes that were more optimal than the ones shown in the example, it would serve as an indication of success. We would be able to have evidence of the adjusted model producing a more optimized output. More thorough testing would need to be completed in order to fully determine whether the adjusted algorithm is better than the original, and this meant incorporating other sample datasets. We would have to run the original and the adjusted models on the different sample datasets and compare the resulting routes. It would be especially important, before making the claim that the adjusted algorithm was in fact an improvement to the original, to include larger sample sizes in order to determine whether the adjusted algorithm is scalable.

4. Results

Unfortunately, due to the significant runtime issues mentioned in section 2.3, our results were inconclusive regarding the improvements to the original algorithm. Training the model was incredibly time consuming and the results were either worse or identical to the output of the original algorithm. The following example images demonstrate the best result that the adjusted algorithm was able to output, which was identical to the original algorithm output. These represent the two models tested on a CVRP problem with 100 nodes, the same data that is used for the figures shown in section 2.2. The image on the left is from the original algorithm using the author's model, and the image on the right is from the adjusted algorithm using our model.





Continuous development is being made and for the latest updates on our model, please see the `README` file located in our root directory.

5. Discussion

Our interpretation of these results shows just how complex these routing problems can be. It can take an immense amount of time to even just provide small enhancements to these models, for a multitude of reasons. For someone with little prior knowledge, it can take a long time to do the background research necessary to have the appropriate contextual knowledge to even understand the algorithm being implemented. Moreover, these algorithms are all built off of each other, and so there can be what feels like a deep "rabbit hole" of algorithms and published works that must be read and analyzed in order to understand the design decisions made by the authors of the paper that we ultimately chose to implement.

In order to make significant changes to these algorithms, there must be a plan for not just what to adjust within the code and where, but more importantly how to go about doing so time-effectively. Particularly this is the case when there is a time-constraint on the development of the algorithm, although this is likely always the case. We did not have a clear plan from the outset on how to adjust the code and spent more time than we should have continuously re-training the model after adjusting code that would ultimately prove to have no impact on the resulting output. There are times when making small changes and constantly rebuilding code is helpful for development and can further one's understanding of the codebase overall. In this case, it was ineffective to make countless minor changes to the model and re-build/re-train it after each one, due to the hours-long runtime. A better approach it seems would be to make a few particular meaningful adjustments that are very very likely to benefit the algorithm, and thus methodically reduce the number of times that the model has to be re-trained.

References

Chen, L., Shang, S., and Guo, T. Real-time route search by locations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):574–581, Apr. 2020a. URL https://doi.org/10.1609/aaai.v34i01.5396.

Chen, L., Shang, S., Yao, B., and Li, J. Pay your trip for traffic congestion: Dynamic pricing in traffic-aware road networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):582–589, Apr. 2020b. URL https://doi.org/10.1609/aaai.v34i01.5397.

Delarue, A., Anderson, R., and Tjandraatmadja, C. Reinforcement learning with combinatorial actions: An application to vehicle routing. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 609–620. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/06a9d51e04213572ef0720dd27a84792-Paper.pdf.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

Lu, H., Zhang, X., and Yang, S. A learning-based iterative method for solving vehicle routing problems. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe1334YDH.

Nazari, M., Oroojlooy, A., Snyder, L., and Takac, M. Reinforcement learning for solving the vehicle routing problem. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/9fb4651c05b2ed70fba5afe0b039a550-Paper.pdf.

Yang, G. and Nikolova, E. Approximation algorithms for route planning with nonlinear objectives. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016. URL https://ojs.aaai.org/index.php/AAAI/article/view/10398.