Heuristic Algorithms for Bike Route Generation

Aidan Pieper   Matthew Anderson (Advisor)

Computer Science Department, Union College

Abstract

The problem of generating routes for recreational cyclists can be modeled using the Arc Orienteering Problem (AOP), a known NP-Hard problem. In order to achieve fast response times, previous literature solves the AOP using heuristic algorithms such as Iterated Local Search (ILS). This research implements and analyzes two existing ILS algorithms for bike routing using an open source routing engine called GraphHopper. We propose ILS variants which our experimental results show can produce better routes at the cost of time.

Problem Statement

We want to generate a preferable bike route whose distance is within some budget. In the context of the AOP, our road network can be modeled as a graph where each road is an edge with a cost (e.g., distance) and a score (e.g. a number denoting the bike safety of the road). The goal is to produce a route such that the total accrued cost is less than the budget and collected score is maximized.

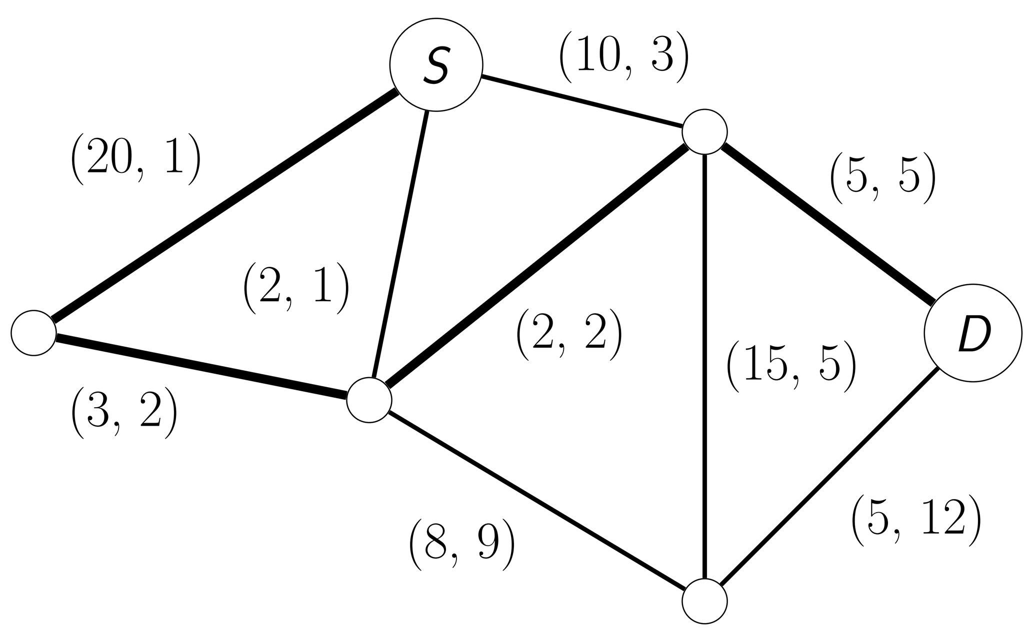


Figure 1: AOP instance with start node S and destination D. Arc label is (score, cost). Bold path is optimal for a budget of 10 (score = 30, cost = 10).

Question

To what extent can ILS algorithms be improved to generate better bike routes?

Related Work

Since known exact algorithms for the AOP are slow on large graphs, previous literature focuses on heuristics such as ILS. ILS uses a search heuristic to iteratively build a sequence of locally optimal solutions [2].

Algorithm 1: ILS(t, $search$, $score$)

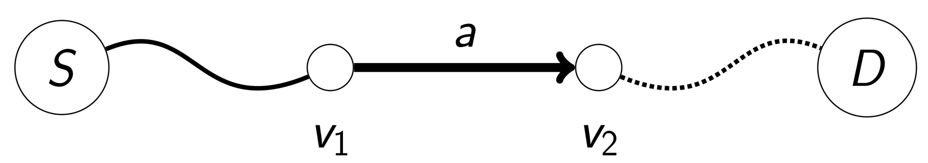
Data: t: a time, $search$: a heuristic search function, $score$: an objective function.

Result: A solution of the search function.

```
S ← search(empty solution);
while t seconds have not elapsed do
    S* ← perturb S;
    S' ← search(S*);
    if score(S') > score(S) then
        S ← S';
return S
```

Our research focus on the following ILS algorithms:

VVA Algorithm: Uses a modified DFS as the local search algorithm and precomputed all-pairs shortest path for feasibility checking [4].



$$(S \to v_1).cost + a.cost + ShortestPath(v_2, D) \leq Budget$$

Figure 2: VVA - arc feasibility checking [4]

LS Algorithm: Reduces the search space by utilizing spatial techniques to prune the number of arcs to search [3].

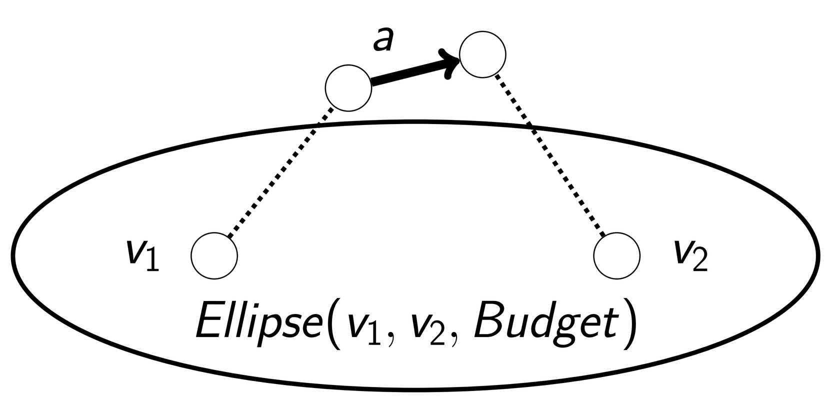


Figure 3: LS - ellipse pruning technique [3]

Methods

We implement the VVA and LS algorithms using an open source routing engine called GraphHopper. GraphHopper downloads and parses OpenStreetMap (OSM) data, a crowd-sourced open mapping dataset, into a usable graph representation for routing algorithms [1].

In our implementation, road costs are determined by distance and road scores are determined by metadata from OSM using GraphHopper's built in bike routing profiles.

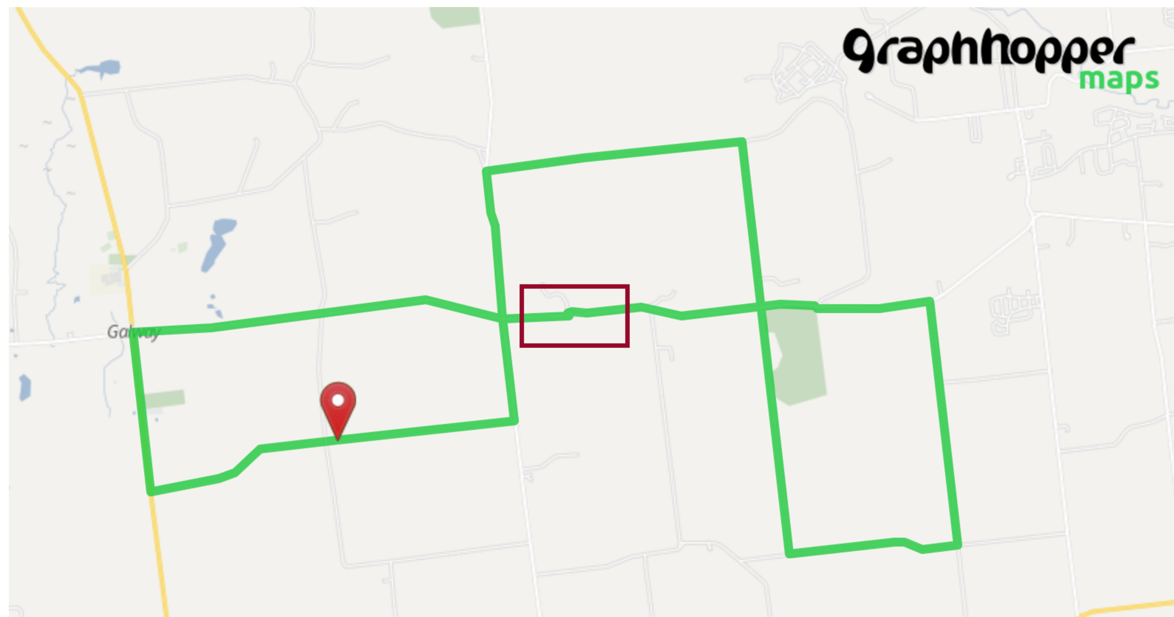


Figure 4: GraphHopper web frontend with generated bike route using VVA. Dangerous turn is highlighted in red.

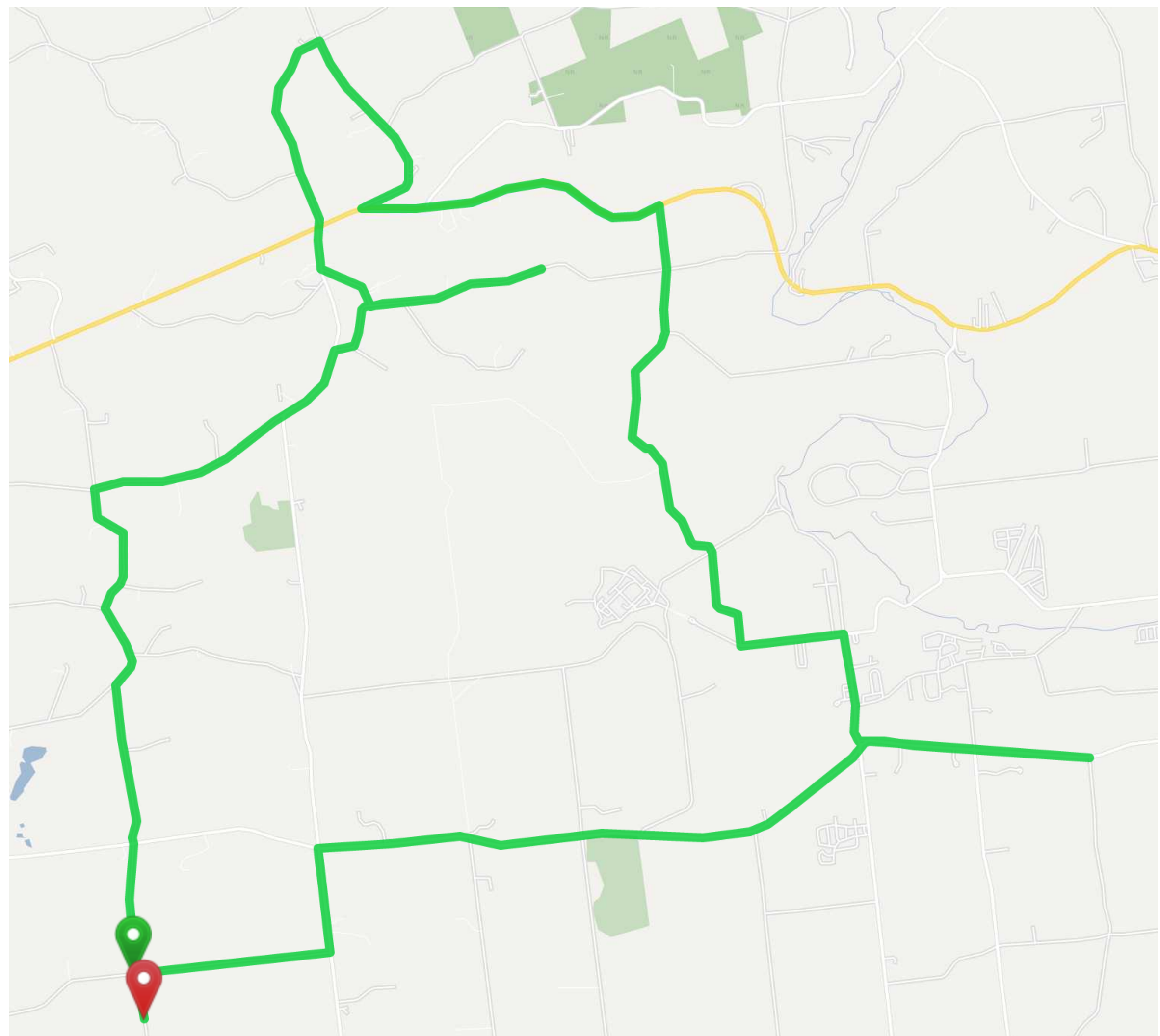


Figure 5: Example route generated with our LS implementation. The route contains some "backtracking" with U-turns.

Our Variants

Based on observations from our LS implementation, we propose the following variants:

- **Budget Allowance & Incremental Budget:** Save distance budget for bigger improvements in later iterations.
- **Arc Restrictions:** Change how roads are chosen.
- **No Backtracking:** Avoid already taken roads when generating route.

Data & Conclusion

We ran a series of experiments to evaluate the performance of the VVA algorithm, the LS algorithm, and our LS variants. Route score and elapsed time were recorded at each iteration.

Algorithm	Score	Time (s)
VVA	19.28	1.01
LS	113.93	0.67
LS + (Budget Allowance)	192.95	10.39
LS + (Incremental Budget)	1.14	1.29
LS + (Arc Restrictions)	49.92	0.06
LS + (No Backtracking)	33.37	0.61
LS + (Budget Allowance) + (Arc Restrictions)	30.80	0.88

Table 1: Algorithm performance with score-cutoff

Our results show that spatial techniques may not drastically speed up the search when using a smart ILS implementation. However, the heuristics used by LS do lead to much higher scoring routes compared to VVA. Some of our LS variants produce even higher scoring routes at the cost of time.

References

[1] Graphhopper routing engine. https://github.com/graphhopper/graphhopper. Visited Nov 2, 2017.

[2] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*, volume 2. Springer, 2010.

[3] Ying Lu and Cyrus Shahabi. An arc orienteering algorithm to find the most scenic path on a large-scale road network. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 46. ACM, 2015.

[4] Cédric Verbeeck, Pieter Vansteenwegen, and E-H Aghezzaf. An extension of the arc orienteering problem and its application to cycle trip planning. *Transportation research part E: logistics and transportation review*, 68:64–78, 2014.