Optimizing Hiking Routes via Integer Linear Programming

Chris Jones
MS Systems Engineering
Student Virginia Tech
Arlington, Virginia
chrisjones@vt.edu

ABSTRACT

This study attempts to identify an optimal strategy to traverse a large network of hiking trails using an Integer Linear Programming formulation and subject to a variety of constraints.

The paper focuses on the “side-to-side” network of 88 hiking trails that are all connected to the Long Trail in Vermont, but discusses scaled down “toy” problems that were used to test model development and addresses applicability to other hiking trail networks.

The simple most basic version of the problem is to hike the entire network in a single trip and can be formulated as the classic “Chinese Postman Problem” which is well-researched and easily solved in polynomial time using commercially-available ILP solvers.

Several additional formulations and extensions of the problem are proposed and explored that include additional constraints that might apply to real-world hiking trips, but future study will be required to solve some.

ACM Reference format:

Chris Jones. 2020. Optimizing Hiking Routes via Integer Linear Programming. In *<<PUBLICATION>>.* Arlington, Virginia

1 Introduction

Postman problems are a set of classic graph theory problems that require finding an optimal tour through a graph that traverses a set of edges. The original description of the problem described a postman who needed to deliver mail to homes on every street in a town, but this same problem has been shown to be relevant to several other applications ranging from snow removal to website usability research [1]. The goal of this project is to propose a new application of postman problems to find the optimal path to traverse a network of hiking trails.

The graph for this analysis was constructed based on the Side-to-Side trail network maintained by the Green Mountain Club of Vermont.

2 Problem Motivation

The Green Mountain Club (GMC) of Vermont maintains and protects the Long Trail (LT), the oldest long-distance hiking trail in the United States. The long trail extends 273 miles along the high spine of the state of Vermont, from the Massachusetts border to the Canadian border, and is co-located with the Appalachian Trail for 105 miles in southern Vermont. Hikers completing the entire Long Trail are known as End-to-Enders.

In addition to the Long Trail, the GMC maintains 89 side trails totaling approximately 167 miles. These side trails lead from roadside parking areas to the Long Trail, from the Long Trail to special features along the way, or in some popular recreational areas they comprise an interconnected network to enable hikers to explore unique wilderness areas. Dedicated hikers can attempt to complete all the side trails to complete a Long Trail Side-to-Side (S2S) hike.

Because of the unique and remote trail system represented by the 89 side trails in the S2S network, hikers wishing to complete a S2S hike must walk far more than the required 167 miles. Many trails must be walked in both directions, while others may be walked in a circuit with other side trails, the LT, or even rural roadways. The hiker attempting to truly minimize the miles walked or the time expended would face

a complex logistics problem. It is this unique and interesting optimization problem that motivates this study.

While the GMC's procedure to become a "Certified Side-To-Sider" requires only traversing the side-to-side trails themselves, practical applications might lead to hikers approaching this goal with several additional sets of constraints:

1. In a single trip as part of a Long Trail thru hike
2. In a single trip, but only using the parts of the Long Trail that are necessary to complete the S2S trails
3. In multiple "tours" that can each start and end at a single parking lot and can be completed in a single day (we estimated that a fit dedicated hiker can travel 20 miles in a single day)
4. In multiple single-day "tours" that can start and end at different parking lots. This can be achieved with a hiking partner (by leaving one car at the end lot) or with a bicycle.

This paper presents a generalizable formulation that can be adapted to any hiking trail network and addresses each of these four problems but focuses on problem formulations 2 ("the single-tour problem") and 3 ("the multi-tour problem").

3 Description of Process

3.1 Data Collection

Digital data for the LT and S2S trail networks were not readily available, so paper maps were purchased from the GMC website. The maps were scanned and combined into a single map on a shared axis and scale using digital photo editing software. A simple Python utility was used to display the combined trail photo, tag each trail and intersection, and convert it into a graph representation where trails are represented by edges and their intersections are represented by nodes (see Figure 1).

The Graph was augmented with the following data:

- Trail lengths for each segment in the LT and each S2S trail retrieved from the GMC website
- For some tours it may also be necessary to walk on roadways, so several key roads were identified and added
- Each node was tagged with a binary attribute identifying whether it included parking (ie whether it can be used as a trailhead to start a tour)

To help with model formulation, several additional "toy" graphs of various complexity were developed (for example, see Figure 2).

Figure 1: **The Long Trail and all trails in the Side-to-Side network, as represented by GMC paper maps (scaled and rotated to form a single map) and in the final graph representation.**

3.2 Model Formulation

The complete graph constructed in section 3.1 is represented as a directional graph $G = (V, A)$ where V is the set of nodes representing the intersections between edges (referred to here as hiking "trails"), and A is the set of trails. Each directed arc in the graph is defined by the trail's southernmost node, northernmost node, and the arc's

direction[1] $(u, v, d) \in A$ and has a cost based on its walking[2] distance, c_{uv}. The trails are approached via T tours (for the single-tour problem, $T = 1$). For each tour, $t \in \{1...T\}$, and for each edge $(u, v, d) \in A$, we define a binary variable x_{uvdt} which takes the value 1 if edge (u, v) in direction d is included in tour t, and a value of 0 if not.

For example, the variable with $u = 1$, $v = 2$, $d = 0$, $t = 0$ is the binary variable that represents whether or not the Broad Brook Trail is traversed in the northbound direction as part of tour number 0.

3.3 ILP Formulation

The hiker's goal is to minimize the total distance travelled while completing the all of the trails in the side-to-side network.

In the ILP formulation, the objective function is to minimize the total distance of all arcs that are travelled across all tours and can be expressed as:

Minimize $$\sum_{(u,v,d)\in A, t\in\{1...T\}} c_{uv} x_{uvdt}$$

The formulation includes several constraints:

- Every variable is binary. This includes an assumption that there is no case where the hiker would need to traverse the same trail in the same direction twice.
  $$0 \le X_{uvdt} \le 1, \ \forall (u, v, d) \in G, \ (t) \in \{1...T\}$$

- It must be possible to start each tour from a parking lot. A single node (999) representing "the road" was defined and for every node that was identified as a parking lot in section 3.1, an edge[3] was added to node 999. Each tour must include node 999.
  $$\sum_{(u,d)\in G,\, v=999} X_{uvdt} \ge 1, \ \forall (t) \in \{1...T\}$$

- The side-to-side network is a proper subset[4] of the full graph ($S \subset G$). For each trail in the side-to-side network, it is required to traverse the trail at least once over the course of all tours (in any direction). So for each undirected edge (u,v) in the side-to-side network, the total value of all X with that (u,v) be >0
  $$\sum_{(u,v)\in S} X_{uvdt} \ge 1$$

- Each tour must not exceed 20 miles. So for each value of t, the total hiking distance of all variables containing that value must be <=20
  $$\sum_{(u,v,d)\in G} C_{uv} X_{uvdt} \le 20, \ \forall (t) \in \{1...T\}$$

- Each tour must be Eulerian. A Eulerian tour requires (and will always exist if) no more than two nodes are of odd degrees (because the two odd nodes are the start and end node). For now we are limiting to tours that start and end at the same node, so every node $u \in V$ must be even for every tour. So for every node in every tour, the total value of all variables adjacent to that node must be even[5]. This constraint is implemented by introducing an integer i
  $$\exists i \in Z \ \forall (u, t) \in (V, \{1...T\})$$
  $$\sum_{(u,v,d)\in G} C_{uv} X_{uvdt} = 2i$$

- In the multi-tour problem, it is necessary to add additional subtour elimination constraints. See section 4.2 for a discussion of these constraints.

[1] Note that edges in directed graphs are traditionally defined using only start and end nodes. To make the programming easier, I chose to name all edges with the southernmost point first and then add another notation for walking direction. For example. Instead of having two edges called "(1,2)" and "(2,1)", I represented the edge as "(1,2,northbound)" and "(1,2,southbound)". So in traditional directional graph notation the directed arc "(2,1)" would be the represented here as "(1,2,southbound)". This formulation makes it easier to express the two separate types of constraints – the GMC thinks of the trails as undirected (they only care that hikers traverse each trail once in any direction), but hikers on the trails think of them as directed.

[2] Note that walking distance applies to all trails and all roads that the hiker might need to walk on. For example the solution in Figure 2 requires walking 1.3 miles on a road. See section 3.3 for a discussion of driving distances.

[3] To avoid solutions that included too much unnecessary driving, roads were given very small (but nonzero) cost.

[4] The long trail, roads, and some other trails are not in the side-to-side network and so are not required, so they have no such constraint. The version of the Postman Problem that includes optional edges is often referred to as the "Rural Postman Problem"

[5] This "evenness" constraint must be true both for walking and for driving, so each node in each tour has two similar constraints – one for driving and one for walking.

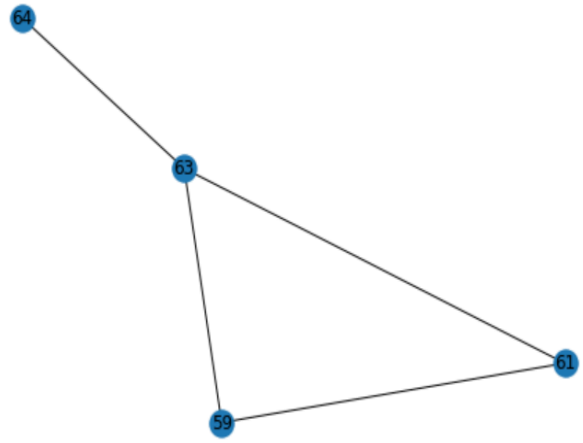

Figure 2: **Example of a "Toy" graph representing the Shrewsbury Peak Trail and Black Swamp Trail. From this example it is clear that optimal tour for this network requires parking at either node 59 or 61, and traversing the edge (63, 64) twice, resulting in a total distance of 9.5 miles.**

4 Results

The ILP formulation described in section 3.3 works well for the single-tour problem. The model and constraints were developed in Python and Gurobi was used to solve the resulting ILP.

Gurobi developed optimal solutions for several toy problems of increasing complexity (Figure 2 is an example) and then on the full graph.

For a single tour that connects all the trails in the S2S network (problem formulation 2), the best solution would take 295 miles. To also include the entire LT (problem formulation 1), the best solution would take 572.6 miles.

4.1 Multi-Tour Solution and subtour elimination

At the time of this paper submission, a solution to the multi-tour solution is not complete.

When extended to multiple tours, the existing constraints return solutions that are infeasible because they contain multiple separate trail segments that a hiker could not possibly traverse together (see Figure 3).

The solution to the multi-tour problem requires implementing an additional set of constraints to resolve and eliminate infeasible solutions. The number and variety of these constraints is large enough that these constraints can not be specified in the model definition phase, so the following iterative solution is used:

1. Define and execute the base model
2. Retrieve the optimal solution
3. Examine the optimal solution and determine whether it is feasible. If it is infeasible, determine an additional inequality that disqualifies the solution, add it to the model, and then re-optimize
4. Repeat until a feasible solution is found

Figure 3 shows two possible constraints that could be introduced to address infeasible tours:

- For tours that are almost entirely connected in a single primary subtour[6] but contain just a few disconnected and distant edges, introduce a constraint to exclude those edges
    - For similar tours where the disconnected edges are physically close to the primary subtour, excluding may not be appropriate (in the optimal answer the primary subtour may expand to encompass the additional edge).
- For tours where no single primary subtour can be identified, an XOR constraint can be created using the binary values of the binary variable x_{uvdt}. For a pair of edges that come from distant parts of the graph, this constraint effectively says "this tour may contain either one of these subtours, but it may not contain both"

[6] The definition of "Primary Subtour" requires tuning. The current implementation defines a primary subtour as having ≥4 nodes, but alternative implementations might involve total distance.

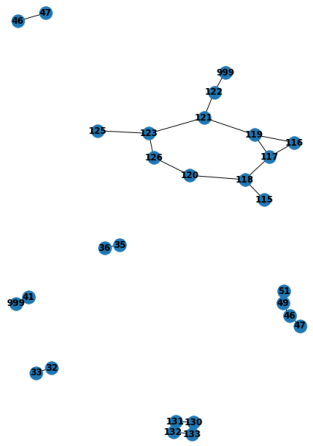

Figure 3: Two impossible tours produced during the first run of the multi-tour problem. The tour on the left has a clear "primary subtour" but has a disconnected and distant subtour that must be eliminated. The tour on the right has no clear "primary subtour" and might be a good candidate for an XOR constraint.

The current implementation takes the number of tours as an input. Once the subtour elimination constraints are solved, additional logic should be added to iterate over multiple different numbers of tours to identify which number of tours results in the shortest total walking distance.

5 Alternative Implementations

Although not yet complete, the additional constraints described in section 4.1 should eventually result in a feasible solution for the multi-tour problem. This study does not investigate whether that solution will be optimal, and it is possible that the process of iteratively adding constraints will result in a feasible but sub-optimal solution.

Alternative implementations to solve the multi-tour problem might include a hybrid algorithm that iteratively uses graph cutting or clustering algorithms to group the trails into tours and then uses the single-tour ILP formulation to solve the optimal route through each tour.

Additional study would be required to research those implementations and their relative strengths and optimality versus a pure ILP formulation.

6 Possible Extensions

Beyond the alternative implementations discussed in section 5, there are several extensions to this problem that would present interesting opportunities for future study.

We did not yet investigate problem formulation 4 as described in the Problem Motivation section (Where the hiker can start and end each tour at different trailheads, either with the help of a friend or by caching a bicycle ahead of time). Once the multi-tour problem is solved, this version would be a simple relaxation of that solution that allows for Eulerian tours where two graph nodes in each tour have odd degree (the start and end node).

For the output of this tour to be practically useful for an actual hiker, it would be valuable to capture additional data about the difficulty of each trail. This formulation states that the hiker can travel 20 miles per day, but experienced hikers know that 20 miles on some trails take far longer than 20 miles on other trails. If better data on trail difficulty and hiker ability was available, travel time would be a better constraint than distance.

Future studies could also consider objective functions that consider the number of tours in addition to the total distance – this objective function may be valuable to a hiker who wishes to minimize the number of vacation days that they need to take from their work or who wishes to maximize the amount of trails that they can cover within a limited amount of time.

The practically useful answer would also sequence the tours so that tours on sequential days are near each other. The current implementation ignores this constraint by assuming that all driving is free (the only objective is to minimize walking distance). Optimal tour sequencing might also consider fatigue and attempt to schedule easier tours on days after harder tours.

7 Conclusions

Although this project has not yet successfully implemented an ILP solution for the multi-tour problem for the S2S trail system, it shows promise as an effective solution to the problem.

This paper also addresses some of the challenges with ILP formulations, introduces subtour elimination constraints that did not appear in other articles on the topic, and explores possible extensions for future study.

There is nothing specific about the model formulation that makes it unique to the S2S trail network, so the model presented here can be applied to any trail network – the author is already using it to plan an upcoming backpacking trip to the Monongahela National Forest in West Virginia.

REFERENCES

[1] H. Thimbleby, "The directed chinese postman problem," *Software: Practice and Experience,* vol. 33, no. 11, pp. 1081-1096, 2003.