Optimizing Grooming of Ski Trails

Grisha Hatavets
Advisor: Sally Cockburn

June, 9 2023

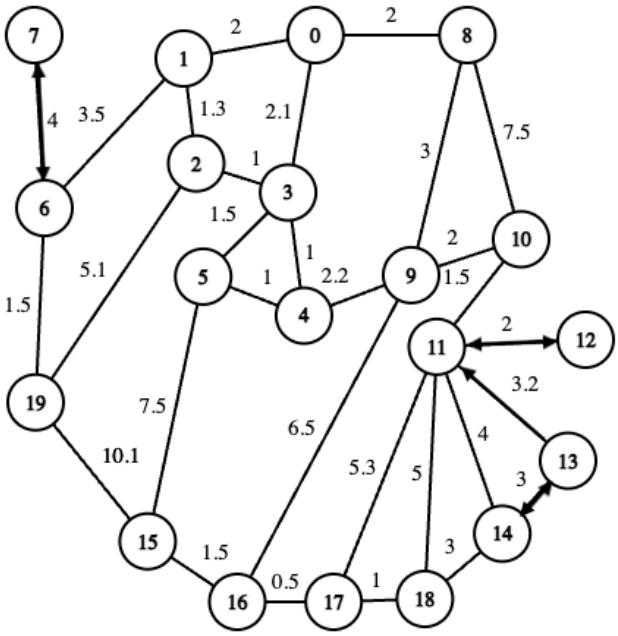
1 Introduction

Every winter, a few faculty members at Hamilton take part in grooming the ski trails (Nordic Trails) on campus. I was approached by them with a project to minimize the total distance covered in order to groom the whole trail (see map below).

This project was modeled as an Integer Program which and solved using Gurobi optimizing software.

2 Model

We begin by defining an edge as a part of the trail which starts at some intersection (vertex) of trails on the map and ends at another intersection. We also add a couple more vertices and edges to account for the possible loops in our map.



The graph shows the distances between vertices and the instance of the problem. An edge with two arrows at its endpoints is a *loop*, and the edge with one arrow can only be covered in the same direction of that arrow. An edge without arrows can be covered either way as long as the same edge is not covered twice in any two consecutive steps, therefore eliminating u-turns in our path. Our goal is to find a directed path covering all of the edges at least twice (and some more than twice) with minimum distance.

2.1 Variables

We define E as the set of all edges on the map, and V as the set of all vertices. For each vertex $i \in V$, let $N_G(i)$ be the set of vertices i is adjacent to. We also define the maximum number of possible steps in our path as M. For this instance, $M = 100$ was a reasonable choice; but, in general, manually finding the number of steps in a couple possible paths (feasible solutions) and adding some more steps to it (possibly doubling the total number of steps) is enough to get a safe value for M.

For each $i \in V$, $j \in N_G(i)$, $k \in \{1, ..., M\}$ we define the binary variable $x_{i,j,k}$ as follows

$$x_{i,j,k} = \begin{cases} 1 \text{ if the path takes an edge from vertex } i \text{ to } j \text{ at step } k, \\ 0 \text{ otherwise.} \end{cases}$$

2.2 Objective Function

The objective is to **minimize** the distance of the path:

$$z = \sum_{i \in V} \sum_{j \in N_G(i)} \sum_{k=1}^{M} x_{i,j,k} \cdot c_{i,j},$$

where $c_{i,j}$ is the measured distance of the edge from vertex i to j.

2.3 Constraints

The path must start at vertex 0 since that is the closest vertex to the equipment storage. The constraint for this condition takes the following form:

$$\sum_{j \in N_G(0)}^{\infty} x_{0,j,1} = 1.$$

The path must also end at the same starting vertex 0. This can be accomplished by a circuit constraint which sets the number of incoming edges equal to the number of outgoing edges at every vertex:

$$\sum_{j \in N_G(i)} \sum_{k=1}^{M} x_{i,j,k} = \sum_{j \in N_G(i)} \sum_{k=1}^{M} x_{j,i,k}, \text{ for all } i \in V.$$

The constraint that takes care of covering every edge at least twice can be written as:

$$\sum_{k=1}^{M} x_{i,j,k} + \sum_{k=1}^{M} x_{j,i,k} \geq 2, \text{ for all } i \in V, j \in N_G(i).$$

The next constraint restricts solutions from assigning more than one edge to any given step:

$$\sum_{ij \in E} x_{i,j,k} \leq 1, \text{ for all } k \in \mathbf{N} : k \leq M.$$

We also want our model to assign the steps sequentially, and the starting vertex of an edge must be at the terminal vertex of the previous edge:

$$\sum_{j \in N_G(i)} x_{j,i,k-1} - \sum_{j \in N_G(i)} x_{i,j,k} \geq 0, \text{ for all } i \in V, k \in \{2, 3, \ldots, M\}.$$

One of the edges $(11 \rightarrow 13)$ is an uphill road segment which can only be groomed in the opposite direction $(13 \rightarrow 11)$. We can simply restrict our model from taking the wrong way along the path:

$$x_{11,13,k} = 0, \text{ for all } k \in \mathbf{N} : k \leq M.$$

Due to the complications with the equipment taking u-turns on the narrow roads of the map, we want to restrict moving along the same edge at any two consecutive steps in our model for the edges that are not loops [1]:

$$x_{i,j,k-1} + x_{j,i,k} \leq 1, \text{ for all } i \in V, j \in N_G(i) : (i,j) \notin L, k \in \{2, 3, \ldots, M\},$$

where L is the set of directed edges that are parts of the loop (i.e.

$$L = \{(6,7), (7,6), (11,12), (12,11), (14,13), (13,14)\}.$$

Similarly, we want to restrict our path from taking a few sharp turns. One example of such turn is located at vertex 1 where we can not turn to vertex 2 if we came from vertex 6, and vice versa. For this example, this is accomplished by two simple constraints:

$$x_{2,1,k-1} + x_{1,6,k} <= 1, \text{ for all } k \in \{2, 3, \ldots, M\},$$

$$x_{6,1,k-1} + x_{1,2,k} <= 1, \text{ for all } k \in \{2, 3, \ldots, M\}.$$

Similar constraints were considered for other restricted turns but are omitted in this paper.

Between vertices 11 and 14, there are two trails that need to be groomed. For the sake of simplicity in programming the model, we treat those two trails as one edge and modify the distance of that single edge to be the average of the distances in corresponding two edges. We then need to add a constraint which forces the solutions to cover that single edge at least four times (twice for one trail and twice for the other):

[1] An example of a loop is an edge between vertices 6 and 7. The model allows a path to cover the loop by taking a directed edge from 6 to 7 and moving from 7 to 6 right after.

$$\sum_{k=1}^{M} x_{i,j,k} + \sum_{k=1}^{M} x_{j,i,k} \geq 4, \text{ for all } i \in V, j \in N_G(i) : (i,j) \in T_2,$$

where $T_2 = \{(11,14),(14,11)\}$.

Some sections of the map are required to be groomed at least three times. To accomplish this, we let T_3 be the set of edges that are on these sections. Then, the constraint takes the following form:

$$\sum_{k=1}^{M} x_{i,j,k} + \sum_{k=1}^{M} x_{j,i,k} \geq 3, \text{ for all } i \in V, j \in N_G(i) : (i,j) \in T_3.$$

One loop, which is required to be groomed three times, only contains two vertices (13 and 14). To keep things simple again, we keep one edge between the vertices and adjust the distance for that edge to be the average of two different paths we can take along the loop. Passing this loop once would mean going from either 13 to 14 or from 14 to 13 and coming back to where we started, thus traversing the edge between 13 and 14 twice. Passing this loop three times would mean traversing the edge between 13 and 14 six times:

$$\sum_{k=1}^{M} x_{i,j,k} + \sum_{k=1}^{M} x_{j,i,k} \geq 6, \text{ for all } i \in V, j \in N_G(i) : (i,j) \in T_6,$$

where $T_6 = \{(13,14),(14,13)\}$.

The path was also required to cover every edge at least twice in the same direction. For every edge e, we introduce a binary decision variable

$$y_e = \begin{cases} 1 \text{ if condition } \mathbf{b)} \text{ is active for edge } e, \\ 0 \text{ if condition } \mathbf{a)} \text{ is active for edge } e. \end{cases}$$

The condition then takes the form of the following two constraints:

$$\mathbf{a)}\ M * y_j + \sum_{k=1}^{M} x_{i,j,k} \geq 2, \text{ for all } (i,j) \in E,$$

$$\mathbf{b)}\ M * (1 - y_j) + \sum_{k=1}^{M} x_{j,i,k} \geq 2, \text{ for all } (i,j) \in E.$$

These last two constraints are optional and give us an ideal grooming path. Groomers asked for a path with and without the last condition to compare the results, so we added the last two constraints after retrieving a directed path without them.

3 Results

We were initially working with a different map which has a scale factor to calculate real values for distances between the vertices. However, after finding a

few differences in the trails of the two maps which could have affected the way vertices were assigned, we decided to work with the map shown in this paper. This map does not have any scale factor, so the distances we measured do not have any real value. As long as the proportions of the real values are maintained in the measurements, we are guaranteed the same optimal paths. However, we can not give any real value to the objective value without knowing the real distance of at least one edge. This and the accuracy of our measurements are the possible limitations in our work.

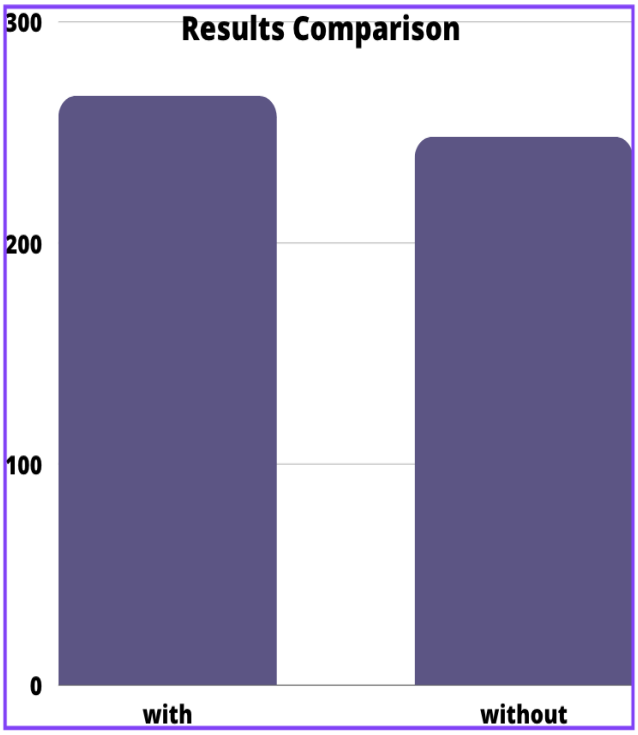
Without the last two constraints, the program was running for approximately 2.5 hours before we forced it to stop. During this time, it found the solution with an objective value of 247.7[2] which was within 2.05% of the best bound. It is worth noting that the solution remained unchanged for the last hour of running the program. The optimal solution was found by outputting only the subscripts of the binary decision variables whose values equaled 1.
The optimal path with $k \in \{0, 1, \ldots, 80\}$ is:

$$0 \to 1 \to 6 \to 19 \to 15 \to 16 \to 9 \to 8 \to 10 \to 9 \to 8 \to 0 \to 1 \to 2 \to 3 \to 0 \to 1 \to 6 \to 7 \to 6 \to 19 \to 2 \to 3 \to 0 \to 8 \to 10 \to 11 \to 14 \to 13 \to 14 \to 13 \to 11 \to 18 \to 17 \to 16 \to 9 \to 4 \to 5 \to 15 \to 19 \to 2 \to 3 \to 4 \to 9 \to 10 \to 8 \to 9 \to 10 \to 11 \to 12 \to 11 \to 14 \to 13 \to 14 \to 11 \to 18 \to 14 \to 11 \to 17 \to 18 \to 14 \to 13 \to 11 \to 17 \to 16 \to 15 \to 5 \to 3 \to 2 \to 1 \to 0 \to 3 \to 5 \to 4 \to 3 \to 0 \to 1 \to 2 \to 19 \to 6 \to 1 \to 0.$$

With the last two constraints, the program found the optimal solution within 15 minutes with an objective value of 266.3. As expected, the path took more steps this time to cover every edge at least twice in the same direction. The optimal path with $k \in \{0, 1, \ldots, 90\}$ is:

$$0 \to 1 \to 2 \to 3 \to 5 \to 4 \to 9 \to 10 \to 8 \to 0 \to 1 \to 6 \to 7 \to 6 \to 7 \to 6 \to 19 \to 2 \to 3 \to 0 \to 1 \to 6 \to 19 \to 15 \to 16 \to 9 \to 8 \to 0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 3 \to 4 \to 9 \to 10 \to 11 \to 12 \to 11 \to 14 \to 13 \to 11 \to 17 \to 18 \to 11 \to 12 \to 11 \to 14 \to 13 \to 14 \to 18 \to 17 \to 16 \to 15 \to 5 \to 3 \to 0 \to 1 \to 6 \to 19 \to 2 \to 1 \to 0 \to 3 \to 2 \to 19 \to 15 \to 16 \to 9 \to 8 \to 10 \to 11 \to 14 \to 13 \to 14 \to 13 \to 11 \to 17 \to 18 \to 11 \to 14 \to 18 \to 17 \to 16 \to 15 \to 5 \to 4 \to 9 \to 10 \to 8 \to 0.$$

[2]Since there were multiple instances of a distance (on a scaled version of the map) measured to be near 0.5 cm, the author decided to take this value as a standard for the units of the input. Therefore, the values which were inputted (same values are on the graph) were in half of cm, or, in other words, "ducentimeter".



There are certainly multiple optimal solutions that Gurobi found for both cases. This solutions differed in the choice of when to take the "loop" edge (i.e $6 \to 7$ or $11 \to 12$). Since both of the vertices that are adjacent to loops have to be visited more than twice, the choice of when to take a loop does not change the potential of objective value at all, as long as the path does not repeat this loop more than what constraints specified.