MATH 381: Project 1

Mimi Aminuddin Hannah An Jack Armstrong Andy Bao

Introduction

Using a set of hiking trails extracted from the Washington Trails Association, each with length, elevation gain, elevation max, rating, and number of rating, our model focuses on determining the best hiking trails to maximize the hiking experience. The model optimal solution is in a form of a subset of hiking trails for the hiker to achieve a minimum goal of total elevation gains. We use binary variables to solve the Linear Programming using the lpsolve. To study the behavior of the optimal solutions, we varied the objective functions and studied the impact of changing few constraints and observe these changes by graphing the results.

Background

We considered a number of ideas and variations of these ideas before deciding on our final problem. anWe have all hiked in Washington, and Jack especially was interested in using data about hikes in Washington from the Washington Trail Association [11] for a problem involving hiking. Articles are published all the time about the "best" hikes in Washington and in Seattle, and we wanted to investigate what the "best" hikes would look like in Washington when subject to various constraints.

To make the problem more concrete we decided to take the idea of "everesting" in cycling and apply it to hiking in Washington, so finding a group of hikes whose combined elevation gains are equivalent to hiking Mount Everest.

A problem similar to ours that was historically addressed is Traveling Salesman Problem (TSP). TSP was first defined and studied by Karl Menger in 1930 [6]. However, the first people to solve this problem were Dantzig, Fulkerson and Johnson in 1954 [9]. TSP is a linear programming model that finds the shorted route to visit a set of cities in the USA and return to the starting city [4]. This Integer Programming problem was solved by using Cutting-Plane algorithms [8]. This problem is similar to ours in that both problems takes distances between two places into consideration; TSP defines the distances between cities as the costs of travel and our problem uses the distances between our origin to mountains in our constraints. One of the differences between TSP and ours is that the objective function of TSP is to minimize the costs while that of ours is to maximize the rating value. Another difference is that our binary variable ends up with either 0 or 1 to indicate which mountains to visit or not while the binary variable in TSP must take 1 on each city in the set.

Another problem that we've found related to ours is the firehouse problem that determines the best selection of sites to place firehouse [7] by using Integer Programming. This problem uses binary variable like ours to represent that if a site is selected, it's 1 and if not selected, 0. It also has a constraint on the total costs set less than a certain budget. This is very similar to our constraint on the elevation gain in that both problems have a upper bound on the sum of costs/gain. One more feature of this problem that is similar to ours is its objective function. This firehouse problem tries to minimize the sum of the distance from district to its selected firehouse while ours maximizes the sum of the elevation gain of its selected mountain. Comparing the objective functions of the two problems is below.

$$\text{Firehouse problem: minimize} \sum_{j=1}^{J} d_{ij}\, x_{ij}$$

$$\text{Our problem: maximize} \sum_{i=1}^{n} eg_i\, x_i$$

where in the firehouse problem n is the number of sites, $\mathrm{d_{ij}}$ is the distance from district to each firehouse, and $x_{ij} = 1$ meaning that the site *i* is selected to assign site *j* for firehouse and $x_{ij} = 1$ not selected, and in our problem, n denotes the number of hiking trails, eg_i is elevation gain of each hike, and $x_i = 1$ being selected and $x_i = 0$ otherwise. Other than these similarities, there is also a difference between the two problems. As the firehouse problem uses two indices *i* and *j* on whether to assign a district to a site, to make sure that it assigns exactly one firehouse in every district, in its constraint it has to set x_{ij} no greater than y_j which has a binary value to indicate whether site *j* is selected or not. This is different from our problem as in our model we don't care whether the mountains are selected from all different areas of Washington, and rather we set the sum of the selected mountains within a certain distance to be 5.

The third problem we've found that is similar to ours is also a knapsack problem that determines the objects to present on the tour [10]. This problem uses the Branch and Bound method with the enumeration tree that measures the set of solutions by dividing them into smaller sets. This problem seems very different from our problem at the first sight, as this one tries to pick objects and our problem tries to pick places to hike. Since the object we are dealing with, which is mountain to hike, is different from the object that this problem has, which is item to bring, it is more likely that the variables that two problems put in their constraints are different; object-tour problem sets maximum limit on the object's weight while our problem sets maximum limit on the elevation gain of the mountain. However, there are more similarities between the two problems than differences. First, both problems' final goal is to pick the best object or place among others. Both use binary variables to indicate whether to select or not. Another biggest similarity between the two problems is their objective function. This tour-object problem maximizes the sum of the value of the object on the tour, and our problem maximizes the sum of the rating of the mountain to hike.

$$\text{Tour-object problem: maximize} \sum_{j=1}^{n} v_j\, x_j$$

$$\text{Our problem: maximize} \sum_{i=1}^{n} eg_i\, x_i$$

where in the tour-object problem *n* represents the number of objects, *v* denotes the value of object, and x_j indicates whether the object is selected to bring or not, and the subscripts in our problems is the same as stated in the previous paragraph. Another similarity we compare actually comes from the difference we stated above. Both problems set the sum of the value that each object or place has in their constraints and put them under their certain upper bound value.

The Model Itself

The model is inspired by the activity called “Everesting”. In short, “Everesting” is a cycling activity where cyclist would take a route with inclines that add up to the height of Mount Everest. Our model use the same concept of adding up the elevation gain to achieve a certain minimum total of overall, but instead of cycling, our subject is hiking. We want our model to pick the best five hiking trails to add up the elevation gain of each trail to achieve the height of mount Everest which is 20000 ft. Since our model only consider a day-hike we constrained our subsets of hiking trails within a maximum of 100 miles of driving, single trail at 8200 ft of elevation gain, and 10 miles of single hike.

Variables in the LP problem are all binary, with each variable standing for a trail: value 0 being not hiking the trail and value 1 hiking the trail. For instance, X_1= 0 would indicate that trail 1 is not climbed. This would be represented as below in our LP.

$$\text{bin } x_i \text{ for all } i \in \{1,2,\ldots,\text{num of trails}\}$$

The objective function has the purpose of maximizing the total rating of the mountains that we are going to climb. Rating data we use (from Washington Trail Association[11]) for each mountain and trail is the average rating from users in WTA [11]. Since rating loses its reliability when the number of people who rated is small, mountains with less than 5 people voting are excluded. We set rating of mountains as an important factor in our LP since ratings are direct feedbacks from climbers. Rating for a mountain in our LP ranges from 0 to 5, with 5 meaning the highest rating for a mountain. We treat maximizing the rating of mountains as optimizing the experience for climbing.

$$\text{Maximizing} \quad \sum_{i=1}^{\text{number of trails}} rating_i x_i$$

For constraints, three main categories of constraints are considered: distance to each mountain from the starting point, elevation gain of mountains, and hiking distance of mountains. Specifically, we use Red Square of University of Washington as a starting point to determine a single distance to each mountain. Any mountain should have a travel distance less than or equal to 100 miles, since a round trip distance of 200 miles is considered as too much time-taking in a single day trip.

With the same reasoning, mountains of hiking distance less than 10 miles are considered applicable for a day hike.

$$HikingDistance_i x_i \leq 10 \text{ for all } i \in \{1,2,\ldots,\text{num of trail}\}$$

The third kind of constraint is on elevation gain of mountains as elevation gain is one of the important factors to consider for enjoyable hikes. For an enjoyable and achieving hike, we decide that a total elevation gain of all five mountains we pick needs more than 20000ft. This is because the distance from the base camp to the peak of Mt. Everest is about 20000ft, assuming that we go to South ridge route in Mt. Everest [1, 5].

$$\sum_{i=1}^{\text{number of trails}} eg_i\, x_i \geq 20{,}000$$

Then, with 20000ft in total, each hike out of five is on average approximately 4000 ft per mountain. Based on Better Health Channel [3], a site by Australian government, 4900 ft to 8200 ft is an intermediate height before altitude sickness. Based on this range of data, this average of 4000 ft seems reasonable and challenging at the same time.

$$\sum_{i=1}^{\text{number of trails}} X_i = 5$$

As heights greater than 8200 ft can cause headache or vomiting for altitude sickness [3], we also set 8200 ft as an upper bound for single trail's elevation gain. Although elevation gain of 8200 ft does not necessarily mean that a person is at a position of 8200 ft high, it serves as an indicator of saying "a climber has climbed the same height of a altitude-sickness-height mountain."

$$eg_i x_i \leq 8200 \text{ for all } i \in \{1,2,\ldots,\text{num of trail}\}$$

Implementation of our LP model is by lpsolver.exe. Java is used to produce the LPSolve format files. Data on mountains are from Washington Trail Association [11]. Some challenges appear in importing a huge amount of data in Excel format into Java. In order to resolve this problem, a built-in csv file processing commands is widely used. The way that these commands work is taking data in Excel format and then taking each column of data as an input array. In that fashion, data can be easily converted into constraints. For instance, all ratings of trails are taken in as an array. Then, we output the line in LPSolve format that maximizes the sum of the multiplications of each rating in the array by its trail binary variable. For example,

Array rating = [4.4, 4.3, 3.2, 4.0 ...]

Output Objective Function: max $4.4X_1+4.3X_2+3.2X_3+4.0X_4+\ldots$

This way of outputting LPSolve format is also applicable for constraints, for instance:

Array elevationGain = [4640, 4000, 5500, 3950, ...]

Output Constraint: $4640X_1+4000X_2+5500X_3+3950X_4+\ldots \geq 20000$

Way of implementing each constraint is similar. For driving distance to certain trail, any mountain should have a travel distance less than or equal to 100 miles are filtered out in Java, instead of adding in as a constraint in LPSolve.

Solution

We used a single Java program to create the necessary LP files, solve them, and interpret the results. To speed up the process of putting the LP file into the LP solver program and interpreting the results manually, we used a Java wrapper for lp_solve to automate this process. The program first outputs an LP file, then once lp_solve produces a solution, information about the selected hikes is printed out based on the values of all the variables from the solution.

Various command line arguments can be given to the Java program to change the various parameters easily without needing to recompile the program each time. These are described in more detail in the appendix.

Data about the trails was obtained from WTA's website, specifically from this endpoint that the website uses to show a map of all trails in the state [11].

The endpoint is a large JSON file of trail data, which was converted into a CSV file for use in our program. The following columns are the data that we use for our model:

- `tid:` A unique identifier for a trail, used to identify trails on WTA's website.
- `name:` The full name of the trail as listed by WTA.
- `rating:` A rating from 0.0 to 5.0.
- `lat:` The latitude of the hike's starting point.
- `lng:` The longitude of the hike's starting point.
- `elevGain:` The elevation gain, in feet, of the hike.
- `length:` The roundtrip length, in miles, of the hike.

Running the program to solve our original problem is as follows, from the command line:

```
java -cp .:trail_lib/* -Djava.library.path=trail_native/ux64
TrailLPGenerator -n 5 -d 75 --single-length-max 10 --single-elev-max
8200 --elev-min 20000 --min-num-ratings 5 --obj 1
```

More details about running the program can be found in the appendix.

Commentary on solution

For the orginal problem we exclude hikes that have fewer than 5 ratings, are not within 75 miles of our starting point, Red Square, or that have more than 8,200 ft of elevation gain. Running the program with these constraints produces the following output as the "best" 5 hikes for "everesting" in Washington:

```
objective: 21.900000000000006
vesper-peak - Vesper Peak
camp-muir - Camp Muir
mailbox-peak-old-trail - Mailbox Peak - Old Trail
mount-dickerman - Mount Dickerman
mount-si-old-trail - Mount Si - Old Trail
```

In a table with the other information about these hikes:

lng	num_ratings	tid	name	rating	elevGain	lat	elevMax	length	value
-121.735	26	camp-muir	Camp Muir	4.3	4640	46.786	10080	8	
-121.675	99	mailbox-peak-old-trail	Mailbox Peak - Old Trail	4.3	4000	47.4674	4822	5.2	
-121.479	11	vesper-peak	Vesper Peak	4.3	4200	48.0371	6214	8	
-121.49	57	mount-dickerman	Mount Dickerman	4.4	3950	48.0538	5760	8.2	
-121.625	16	mount-si-old-trail	Mount Si - Old Trail	4.6	3420	47.4408	3980	7.4	
	209			21.9	20210			36.8	21.9

With the bottom row showing various totals. We can see that the objective value output comes from the sum of all the ratings of the 5 hikes, following our model.

The five hikes selected as a solution to our original problem seem reasonable. All are highly rated above 4.0, and all are strenuous with the minimum elevation gain for a single hike being 3420 ft. Two of the hikes selected, Mailbox Peak and Mount Si, are both very well-known and well-regarded as Washington hides.

With the goal for this list of hikes being to successfully "everest" through hiking while also getting a taste of the hiking in Washington state, this seems like a valid itinerary for a hiker with this goal to follow.

Variations

1. Rescaling the rating using incremented powers

The model gives us the best five hiking trails in order to achieve the total elevation gain goal. However, the model treats the rating as if it is ranked incrementally. Realistically, we know there is a huge gap in preferences when we look at the ratings between value 1.0 and 2.0, in comparison to the ratings between value 4.0 and 5.0, even though both cases have a difference value of 1.0. Thus, implementing squares in our objective function would justify the real value of the ratings. In a way, this additional approach makes our model more sensitive towards the rating preferences.

Our new objective function becomes:

$$maximize \sum_{i=0}^{n} {rating_i}^k$$

Where n is the total number of hikes and k is a power to raise the rating to, e.g. square, cubic, etc.

The optimal solutions (best five selected hiking trails) are shown on the table below. For various ranges of k, the trails selected were all the same 5.

k	Trails Selected	No. of Rating	Rating	Elev Gains	Total Elev Gain	Lengths	Total Length
1 to 10	Camp Muir Mailbox Peak - Old Trail Vesper Peak	26 99 11	4.3 4.3 4.3	4640 4000 4200	20210	8 5.2 8	36.8

	Mount Dickerman Mount Si - Old Trail	57 16	4.4 4.6	3950 3420		8.2 7.4	
11 to 27	Camp Muir Mildred Point Red Mountain (Painted Mountain) Mount Dickerman Mount Si - Old Trail	26 5 5 57 16	4.3 4.6 3 4.4 4.6	4640 3135 4875 3950 3420	20020	8 7 8 8.2 7.4	38.6

We observe that the optimal solution did not change until k = 11 and the new optimal solution remains the same until k = 27 where the values generated by such a large exponent became too large to handle by the solver. The new five best hiking trails now is closer to the maximum total elevation gain compared to before. However, the new total rating actually decreased from 21.9 to 20.9, by the original objective function where the rating was not taken to any power. Thus, the incremented power did not really have a huge or significant effect to the optimal solutions in picking the best hiking trails. However, it does show that the total elevation gain is more efficient.

2. Rescaling the rating using exponential

Another method to rescale the rating preferences is by using exponential in our objective function. We consider two ways to use the exponential:

Excluded the Number of Rating

This method gives similar approach as incrementing the powers as explained above by the definition of the exponential behavior. Thus, our new objective function is:

$$maximize \sum_{i=0}^{n} \exp(rating_i)$$

where n is again the total number of trails.

As expected, the optimal solutions of the selected best five hiking as shown below yields the same result as the rescaling the rating using incremented power from 2 to 10. In both cases, the value of the original function is still 21.9.

Trails Selected	No. of Rating	Total No. of Rating	Rating	Total Rating	Elev Gain	Total Elev Gain	Length	Total Length
Camp Muir	26	209	4.3	21.9	4640	20,210	8	36.8
Mailbox Peak - Old Trail	99		4.3		4000		5.2	

Vesper Peak	11		4.3		4200		8	
Mount Dickerman	57		4.4		3950		8.2	
Mount Si - Old Trail	16		4.6		3420		7.4	

Included the Number of Rating

Now, say we want take into account the number of people rated in our objective function. This method will now consider that a trail with rating 5.0 rated by 5 people is less valuable than trail with rating 4.7 rated by over 100 people. We want to study the new results to see how big of an impact the relationship between rating and number of rating in the model. In this case, we set our new objective function as:

$$\sum_{i=0}^{n} rating\_count_i * \exp(rating_i)$$

Where rating_count is the number of ratings for a certain trail, and rating is the actual score for a trail, from 0.0 to 5.0.

The selected best five hiking trails yield three new trails with in comparison with the previous results if we observe the table below. With this objective function, it took a value of 31050.5.

Trails Selected	No. of Ratings	Total No. of Ratings	Rating	Total rating (original objective function)	Elev. gain	Total Elev. gain	Length	Total length
Camp Muir	26	374	4.3	19.9	4640	20190	8	39.2
Monogram Lake	5		3.8		4675		9.8	
Red Mountain (Painted Mountain)	5		3		4875		8	
Mailbox Peak - Old Trail	99		4.3		4000		5.2	
Lake Serene - Bridal Veil Falls	239		4.5		2000		8.2	

The result has the lowest total rating compared to the other attempted objective function. On the contrary, this approach have the closest value to the total elevation gain goal. This indicates that this method tends towards to actually suffice the total elevation gain compared to getting the best rated of trails, even though the purpose of this approach is to increase the sensitivity in the relationship between rating and number of rating.

Therefore, different approaches toward changing the objective function to make it more efficient to the model serves different purpose to the selected hiking trails. We refer Appendix A to see the significance of each methods.

3. New Constraints of Distance between Each Trails

We now considering new constraints requiring the distance between each selected trail hiked to be greater than a certain value. This creates a radius around each trail that no other trails can fall within. This idea comes from the thought of hiking trails in different regions create more diversity, since mountains that are close together may share similar characteristics or lookout views. For example in the original solution, Vesper Peak and Mount Dickerman are right next to each other, and while both good hikes, more interesting solutions can be found. Thus, we now include distance between trails as one of the constraints in our model. We define a new binary variable y_{ij} to be 1 if trails x_i and x_j are chosen together. To say that we want the distance between trail x_i and x_j to be greater than some minimum value, say 10 miles. We introduce a very large value variable M in order to check the distance between these two selected trails is above the selected miles. Hence, our new constraints look like:

$$y_{ij} \geq x_i + x_j - 1 \ (1)$$
$$y_{ij} \leq x_i \ (2)$$
$$y_{ij} \leq x_j \ (3)$$
$$M\left(1 - y_{ij}\right) + d * y_{ij} \geq 10$$

where d is the calculated distance between the selected two trails, and where i and j will range over all possible pairs of hikes, e.g:

$$i, j \in \left\{0, 1, \dots, \binom{n}{2}\right\}, i \neq j$$

For all the n-choose-2 pairs of hikes. Only the unique pairs need a corresponding set of constraints added to the LP file.

The first three constraints are requiring y_{ij} to be the logical AND of the two x variables. Notice that if $y_{ij} = 1$, the constraint (1) is checking the existence of the selected paired trail to determine whether to check the distance. If trail i and trail j are both selected, then $y_{ij} = 1$, which indicates that the distance must be taken into check in constraint (2) represented by the value d. If neither trail i nor trail j is selected, then $y_{ij} = 0$ which conveys that the distance does not need to be checked. Furthermore, the constraint (2) is checking either the distance satisfy the minimum between the selected two trails. Constraint (2) is similar to a “if/else” condition to validate if the distance between the two selected trails satisfy the minimum 10 miles distance. Notice that since M has a very large value, if $y_{ij} = 0$, then the left hand side of constraint (2) will always be greater than 10. In the LPsolve file, these 4 constraints are added for every possible pairing of hikes. To show the implementation of this new constraints in our LP model, we set M = 10000 and the selected best trails are as shown below:

We also added a constraint on the distances between each hike to the other hikes, with the intent to not just find any 5 hikes in Washington to choose for everesting, but to try to find hikes that are spaced out from one another to get a range of terrain, views, etc. For example in the original solution, Vesper Peak and Mount Dickerman are right next to each other, and while both good hikes, more interesting solutions can be found.

The original problem was solved with this additional constraint, where the distance between each of the chosen hikes had to be above different values The results are below:

Minimum distance between hikes	Objective function	Trails selected	Trail IDs
5	22.5	Three Fingers (4.5) Camp Muir (4.3) The Enchantments (4.5) Alta Mountain (4.6) Mount Si - Old Trail (4.6)	three-fingers camp-muir enchantment-lakes alta-mountain mount-si-old-trail
10	22.5	Three Fingers (4.5) Camp Muir (4.3) The Enchantments (4.5) Alta Mountain (4.6) Mount Si - Old Trail (4.6)	three-fingers camp-muir enchantment-lakes alta-mountain mount-si-old-trail
15	22.5	Three Fingers (4.5) Camp Muir (4.3) The Enchantments (4.5) Alta Mountain (4.6) Mount Si - Old Trail (4.6)	three-fingers camp-muir enchantment-lakes alta-mountain mount-si-old-trail
20	22.4	Three Fingers (4.5) Camp Muir (4.3) The Enchantments (4.5) Mount Ellinor (4.5) Mount Si - Old Trail (4.6)	three-fingers camp-muir enchantment-lakes mount-ellinor mount-si-old-trail
25	22.4	Three Fingers (4.5) Camp Muir (4.3) The Enchantments (4.5) Mount Ellinor (4.5) Mount Si - Old Trail (4.6)	three-fingers camp-muir enchantment-lakes mount-ellinor mount-si-old-trail
30	22.4	Three Fingers (4.5) Camp Muir (4.3) The Enchantments (4.5) Mount Ellinor (4.5) Mount Si - Old Trail (4.6)	three-fingers camp-muir enchantment-lakes mount-ellinor mount-si-old-trail
35	22.4	Three Fingers (4.5) Camp Muir (4.3) The Enchantments (4.5)	three-fingers camp-muir enchantment-lakes

		Mount Ellinor (4.5) Mount Si - Old Trail (4.6)	mount-ellinor mount-si-old-trail
40	22.1	Three Fingers (4.5) Camp Muir (4.3) The Enchantments (4.5) Grand Ridge (4.5) Mailbox Peak - Old Trail (4.3)	three-fingers camp-muir enchantment-lakes grand-ridge mailbox-peak-old-trail
45	22.1	Three Fingers (4.5) Vesper Peak (4.3) Camp Muir (4.3) Round Lake (4.5) Lila Lake (4.5)	three-fingers vesper-peak camp-muir round-lake lila-lake
50	20.8	Three Fingers (4.5) Mailbox Peak - Old Trail (4.3) Mildred Point (4.6) Navaho Peak (4.4) Red Mountain (Painted Mountain) (3.0)	three-fingers mailbox-peak-old-trail mildred-point navaho-peak red-mountain
60	17.4	Three Fingers (4.5) Camp Muir (4.3) Gladys Divide (2.8) Lake Ann (3.4) Miller Peak (2.4)	three-fingers camp-muir gladys-divide-primitive lake-ann-1 miller-peak

Immediately we see that even requiring at 5 miles of distance around each hike results in a very different solution than the original one, due to Vesper Peak and Mount Dickerman being so close in the original solution. Three Fingers, a mountain close to Vesper and Dickerman, was found instead, and was retained in the solutions all the way to having a 60 mile radius around each hike. Three Fingers is just slightly farther north than the other two, so it appears that this slight bit of distance allowed Three Fingers to be just out of range of the other hikes to be retained in the solution.

4. Considering Different Starting Points

Initially, our model only consider the starting point in Seattle. By varying the starting point we want to study the behavior of the LP model in choosing the best hiking trails to achieve the minimum total elevation gain given that they have different feasible hiking trails subset within the travel distance. We use Spokane, Colville, Olympia, Skykomish, and Wenatchee as our scatter starting points. The results as shown below:

Spokane
The result was infeasible outcome due to the restriction in feasible subset of hiking trails. Thus, we relaxed the distance willing to travel until the result is feasible which is at the maximum distance of 190 miles with objective value of 20.4. The selected trails are as follows:

No. of Ratings	Total No. Of Ratings	Trails Selected	Rating	Total rating	Elev Gain	Total Elev Gain	Length	Total Length
11	60	Vesper Peak	4.3	20.4	4200	20075	8	42.8
6		Goode Ridge	3.5		4400		10	
32		Hidden Lake Lookout	4.5		3300		8	
5		Monogram Lake	3.8		4675		9.8	
6		Mount Baring	4.3		3500		7	

Colville
Similar to Spokane, the result was infeasible outcome due to the limited feasible subset of hiking trails with the given constraints. Thus, we relaxed the distance willing to travel until the result is feasible which is at the maximum distance of 160 miles with objective value of 20.0. The selected trails are as follow:

No. of Rating	Total No. Of Rating	Trails Selected	Rating	Total rating	Elev Gain	Total Elev Gain	Length	Total Length
15	65	Carne Mountain	4.1	20.0	3600	20375	8	42.6
7		Desolation Peak	4.1		4400		6.8	
6		Goode Ridge	3.5		4400		10	
32		Hidden Lake Lookout	4.5		3300		8	
5		Monogram Lake	3.8		4675		9.8	

Olympia, Skykomish and Wenatchee

For the other three starting points as stated above, they all yield the same result as our main model which set Seattle as its starting point.

From this outcome, we can conclude that Seattle is a strategic place to get the best hiking experience due to its diverse available hiking trails under these constraints. Not only that it has higher objective value, it's optimal solutions also yield a more efficient subset in achieving the total elevation gain since its total length of hike is shorter.

4. Changing minimum and maximum elevation gains

The original problem had a set goal of 20,000ft of total elevation gain across 5 trails with no upper bound. We introduced variation for this lower bound value and also introduced an upper bound on the elevation gain to find the best hikes in various windows of elevation gain to account for different goals in hiking. For example instead of "everesting" to 20,000ft one might desire a set of trails in the 5,000-10,000ft range. For each window of elevations the objective function can give insight as to which window of elevations could be considered to have the best hiking as far as a ratings are concerned.

This introduces an additional constraint to the model in the form:

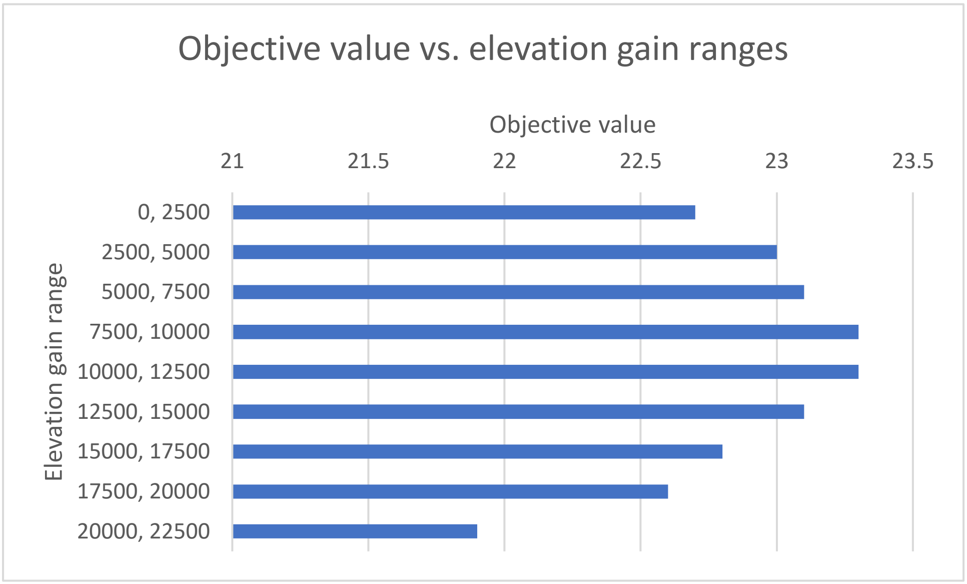
$$\sum_{i=0}^{N} eg_i x_i \leq EG_{max}$$

with EG_{max} being the maximum allowed total elevation gain, eg_i being the elevation gain for trail i, and with the minimum allowed total elevation gain being changed appropriately to create windows of total elevation gain. Various windows are shown in the table below, with the original constraints as in the original problem, other than the variations listed in the table.

Minimum elevation Gain (ft)	Maximum elevation gain (ft)	Elevation gain of selected trails	Objective Function	Trails selected	WTA trail IDs
0	2500	2315	22.7	Chambers Bay Loop (4.8) Mount Erie (4.7) Naches Peak Loop (4.4) Point Defiance Park (4.4) Ranger Hole - Nature Trail (4.4)	chambers-bay-loop mount-erie naches-peak-loop point-defiance-park ranger-hole
2500	5000	4765	23.2	Skyline Trail Loop (4.5) Chambers Bay Loop (4.8) Mount Erie (4.7) Mount McCausland (4.6) Point Defiance Park (4.4)	skyline chambers-bay-loop mount-erie mt.-mccausland point-defiance-park
5000	7500	6420	23.5	Chambers Bay Loop (4.8) Mount Erie (4.7) Mount McCausland (4.6) Snoqualmie Mountain (4.6) Point Defiance Park (4.4)	chambers-bay-loop mount-erie mt.-mccausland mount-snoqualmie point-defiance-park
7500	10000	9355	23.5	Chambers Bay Loop (4.8) Mildred Point (4.6) Mount Erie (4.7) Mount McCausland (4.6) Snoqualmie Mountain (4.6)	chambers-bay-loop mildred-point mount-erie mt.-mccausland mount-snoqualmie
10000	12500	10975	23.4	Chambers Bay Loop (4.8) Mildred Point (4.6) Mount Erie (4.7) Snoqualmie Mountain (4.6) Mount Si - Old Trail (4.6)	chambers-bay-loop mildred-point mount-erie mount-snoqualmie mount-si-old-trail
12500	15000	13275	23.3	Chambers Bay Loop (4.8) Mildred Point (4.6) Mount Ellinor (4.5) Snoqualmie Mountain (4.6) Mount Si - Old Trail (4.6)	chambers-bay-loop mildred-point mount-ellinor mount-snoqualmie mount-si-old-trail
15000	17500	15300	23.1	Camp Muir (4.3) Mildred Point (4.6) Mount Erie (4.7)	camp-muir mildred-point mount-erie

			Snoqualmie Mountain (4.6)	mount-snoqualmie
			Mount Si - Old Trail (4.6)	mount-si-old-trail
17500	20000	22.9	Camp Muir (4.3)	camp-muir
			Mildred Point (4.6)	mildred-point
			Mount Ellinor (4.5)	mount-ellinor
17600			Snoqualmie Mountain (4.6)	mount-snoqualmie
			Mount Si - Old Trail (4.6)	mount-si-old-trail
20000	22500	22.5	Vesper Peak (4.3)	vesper-peak
			Camp Muir (4.3)	camp-muir
			Mailbox Peak - Old Trail (4.3)	mailbox-peak-old-trail
20210			Mount Dickerman (4.4)	mount-dickerman
			Mount Si - Old Trail (4.6)	mount-si-old-trail

Overall trails are largely retained from one elevation gain range to the next, with trails gradually getting pushed out of the top 5 as their elevation gains become infeasible for getting total elevation gain within each of the above ranges. The original problem's solution can be found, as expected in the [20,000, 25,000]. For this variation, no solutions could be found for the next largest elevation gain range, [22,500, 25,000]. Plotting the elevation gain ranges and their respective objective values:



Overall the objective function value slowly increases to a maximum around the ranges [7,500, 10,000] and [10,000, 12,500], and then quickly drops off. The only extreme value is for the range [20,000 - 22,500], the range for the original problem, where the value sharply drops off from the range before. This suggests that less highly rated hikes are found when the minimum elevation gain is required to be a very high value (for the last range, over 4,000 ft gain per hike, on average). This is seen for the total elevation gain in the selected hikes from the table, where for the higher ranges the actual selected elevation gain is very close to the lower bound, whereas for the lower elevation ranges it is closer to the upper bound.

This could be either that very strenuous hikes are not rated as highly as moderate hikes, on average, or that there are fewer combinations of trails that will work when the elevation gain range is very high, which will necessarily limit the value of the objective function. So for true “everesting”, you will not find as highly rated hikes than if you target a more moderate elevation gain window.

Conclusion

We figured out that integer programming problem is useful to figure out climbing plans, and sensitive to constraint changes. Specifically, as in variation parts, manipulation of objective function has a huge impact on the whole line up. At the same time, adding a constraint to the LP problem will also impose change on trails line up easily. Using different logics, different constraints can be achieved. For instance, large threshold value is used in distance between mountains part to achieve if/else conditions.

Overall under many different constraints, unique but still high scoring groups of 5 hikes could still be found around Washington. Without any constraints, of course, the absolute best (according to the objective function) set of 5 hikes will be chosen, and adding additional constraints will necessarily decrease this overall global maximum. However, adding moderate constraints did not significantly decrease the objective value, unless very strict constraints were imposed on the problem.

This suggests that when choosing a relatively small number among a very large group of objects, good solutions still can be found throughout many different constraints on how that small set should be chosen from within the overall set. However, under very strict constraints, for example by requiring very distant hikes to be chosen to see a large part of the state, or by requiring a very large elevation total, the quality of the hikes will necessarily start to drop off.

Measuring the quality of a subset from a much bigger set was difficult to define in this problem. For the example of hiking the quality of hikes might vary significantly from one person to another, and using ratings was an imperfect way to approximate the quality of a set of hikes. For choosing a certain number of hikes from a large set, another measure of quality might be the diversity of those hikes. This was implemented as a constraint in one variation, but adding the notion of trail “diversity” in the objective function may provide better and more interesting result. This could be implemented by, for example, assigning hikes to different categories and assigning higher scores to sets of 5 hikes that represent more of these categories.

When solving a problem using a very large set of objects to choose from, in this case trails, there were instances where the LP solver would seemingly never terminate. For instance when adding the constraint that no two hikes should be closer than a certain distance to each other, adding pairwise constraints across 400+ hikes made the model significantly more complicated. It was pertinent for many of the solutions we generated that the input be sufficiently filtered so as not to clutter the LP solver with hikes that it would likely never choose. For example with a goal of achieving 20,000 ft of total elevation gain across 5 hikes, a valid assumption might be that a hike having 100 ft of elevation gain, even if it was highly rated, does not fit in with other very strenuous hikes, so filtering out these types of hikes can dramatically increase the speed of the solver while not affecting the solutions.

References/Bibliography

[1] Climbing the Seven Summits blog, Anya Zolotusky. Accessed February 2, 2017, http://climbingthesevensummits.com/seven-summits/the-mountains/mt-everest/

[2] Dantzig G., Fulkerson R., and Johnson S. Solution of a Large Scale Traveling Salesman Problem. Technical Report P-510. RAND Corporation, (1954): 393-410. Accessed January 26, 2017, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.9319&rep=rep1&type=pdf

[3] Department of Health & Human Services, State Government of Victoria, Australia. "Altitude sickness." Accessed January 29, 2017, https://www.betterhealth.vic.gov.au/health/healthyliving/altitude-sickness

[4] Diaby, M., "The Travelling Salesman Problem: A Linear Programming Formulation," WSEAS Transactions on Mathematics, Issue 6, Vol. 6 (2007): 745-754.

[5] Everesting. "The Rules" Accessed February 1, 2017, http://www.1]ing.cc/the-rules/

[6] Held, M., Hoffman, A. J., Johnson, E. L., and Wolfe P. "Aspects of the Traveling Salesman Problem," IBM J. Res. Develop, Vol. 28 (1984): 476-486. Accessed February 2, 2017, http://math.mit.edu/~asuk/sales.pdf

[7] Integer Programming. "Some Integer-Programming Models." 285-287. Accessed January 28, 2017, https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjG3teG3fTRAhURS2MKHfsEAxUQFggcMAA&url=https%3A%2F%2Fwww.researchgate.net%2Ffile.PostFileLoader.html%3Fid%3D57d144d2dc332d81575af5ce%26assetKey%3DAS%253A403999657349126%25401473332432828&usg=AFQjCNFAv3Arrah1Wlqtc-ZZVreAdhSy_w&sig2=JEDKmawPttHPSSTl1LEheg

[8] Natural Sciences and Engineering Research Council of Canada (NSERC) and University of Waterloo. "Dantzig, Fulkerson, and Johnson's Cutting-Plane Method." Last modified Apr 2005. http://www.math.uwaterloo.ca/tsp/methods/dfj/

[9] Rocha, Ana Maria A.C., Fernandes, Edite M.G.P., and Soares, Joao Luis C. "Solution of Asymmetric Traveling Salesman Problems Combining the Volume and Simplex Algorithms." (2004): 2, accessed February 2, 2017, http://www.mat.uc.pt/~jsoares/research/ART_ingles.pdf

[10] The Branch and Bound Method. "An Example: the Knapsack Problem." 1252-1259. Accessed February 2, 2017, http://compalg.inf.elte.hu/~tony/Oktatas/SecondExpert/Chapter24-Branch-6April.pdf

[11] Washington Trails Association Official Website. Accessed January 21, 2017, https://www.wta.org/go-hiking/map/@@trailhead-search/getHikes?jsonp_callback=WtaTrailheadSearch.setHikes

Appendix A

Recall that our main model have the same output as the objective function of maximizingi = 1nratingik, where n is the total number of trails and k = 2, 3, 4,..., 10.

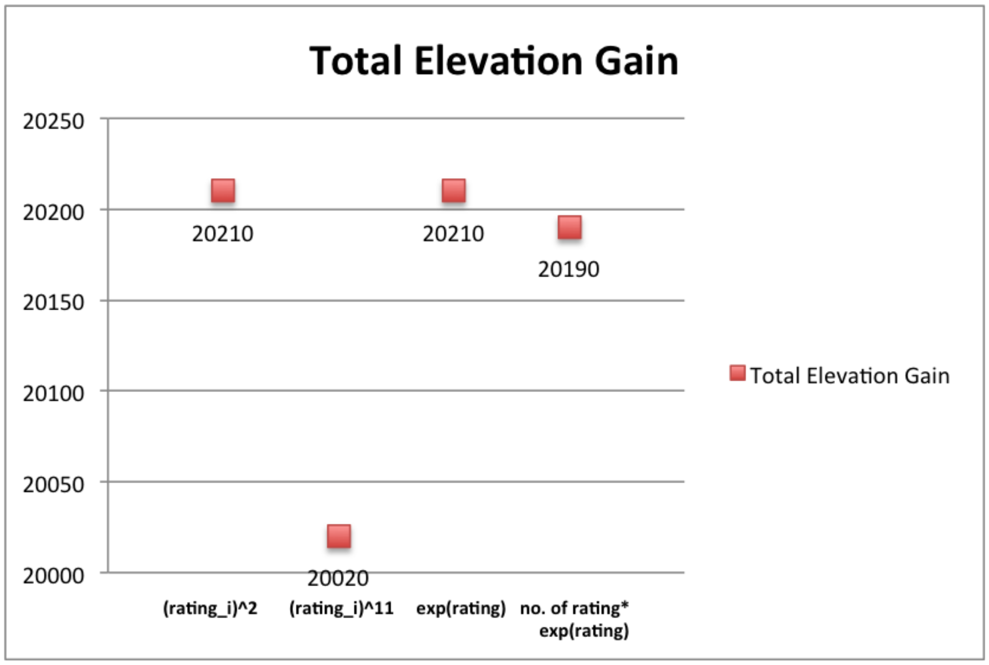


Figure 1: The above figure shows the relationship between the different objective function and the total elevation gain achieved from the optimal solutions (five best selected hiking trails). Recall that we set our minimum goal of total elevation gain to 20,000 ft.

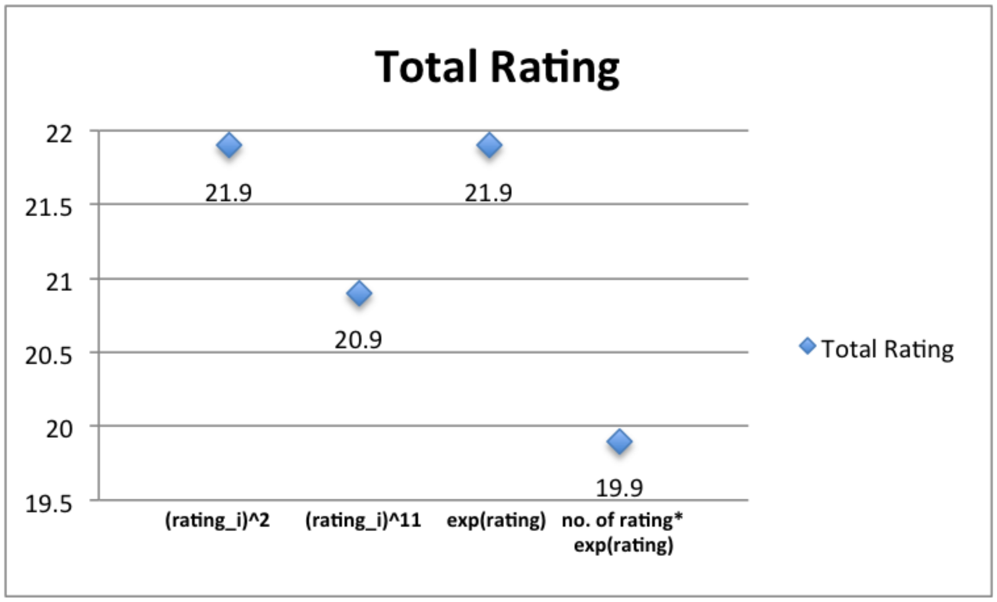


Figure 2: The above figure shows the relationship between the different objective function and the total rating achieved from the optimal solutions (five best selected hiking trails).

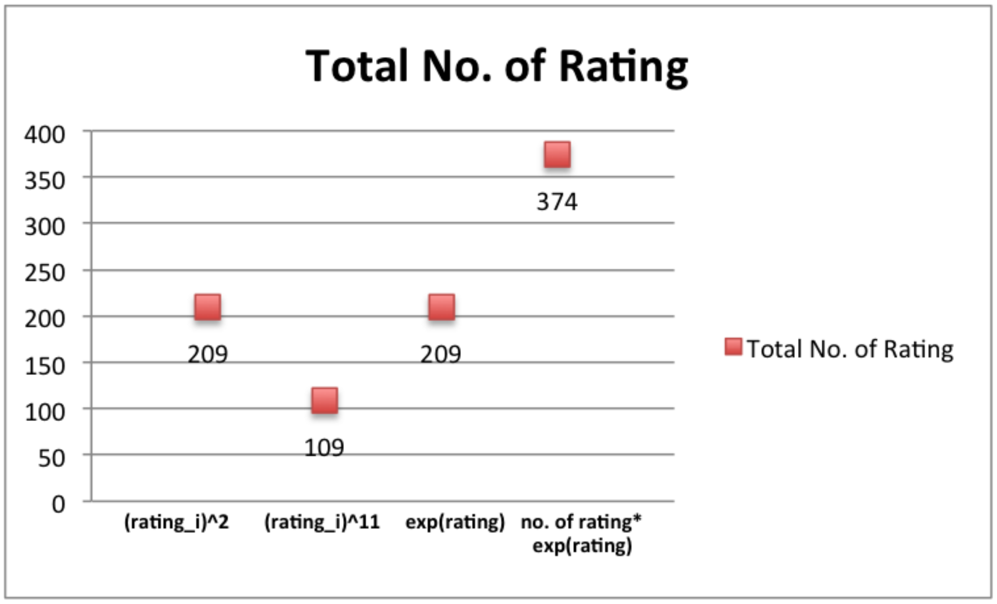


Figure 3: The above figure shows the relationship between the different objective function and the total number of rating achieved from the optimal solutions (five best selected hiking trails). The purpose is to study the reliability of the ratings of the chosen solutions.

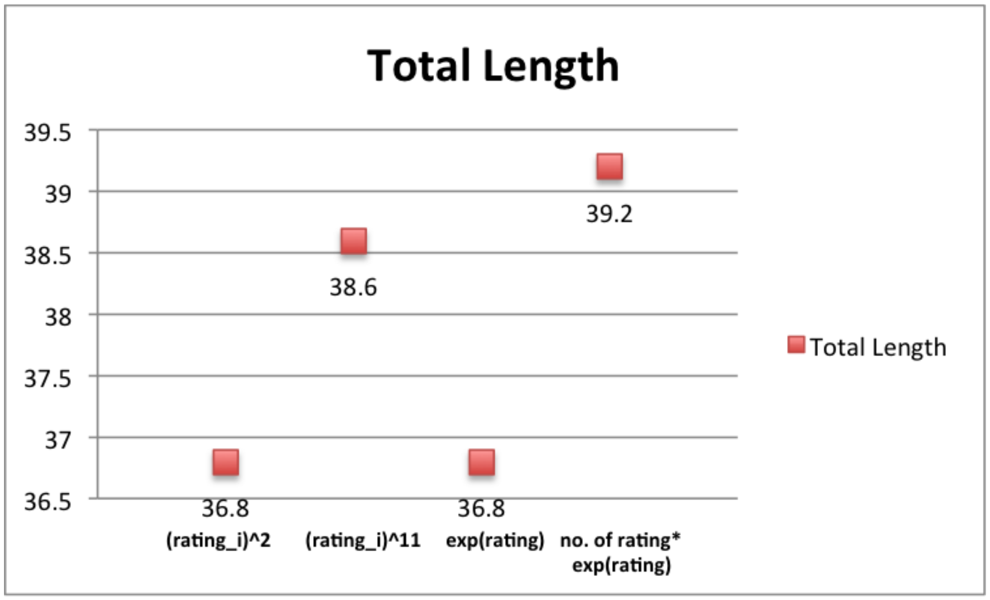


Figure 4: The above figure shows the relationship between the different objective function and the total length achieved from the optimal solutions (five best selected hiking trails). The purpose is to study the efficiency of the selected hiking trails which is to be able to achieve the targeted minimum elevation gain with shorter distance.

Appendix B: running the program

We used a Java program to generate LP files with various parameters and then feed these to lp_solve which will print out the optimal number of trails.

The program can be run most easily on the command line (instructions below) with options to set the number of trails, distance constraints, elevation constraints, etc.

The intent of this is not really to produce software that can actually be used, as there is a large laundry list of options that are certainly not very user friendly to change, but rather to allow solutions with different parameters to be generated and examined easily. Changing values by hand, e.g. the elevation constraints, and running the program manually each time and copying the output in order to collect data for our graphs would have been prohibitively time consuming, especially for the larger tables generated in the paper.

<u>How to run:</u>

Make sure you have these files/folders in the folder you want to run the program in. These are all contained in the .zip file included with the submission.

```
trails_with_ratings.csv
Trail.java
TrailLPGenerator.java
trail_lib/
```
- `commons-csv-1.4.jar`
- `lpsolve55j.jar`

```
trail_native/
```
- `mac/`
- `win32/`
- `win64/`
- `ux32/`
- `ux64/`

Also ensure that you have Java 8 JDK installed and on your path, e.g. type "java" and "javac" on the command line and ensure that these programs both exist.

To compile and run:

On Windows (64-bit):

(to compile the files)
```
javac -cp .;trail_lib/* TrailLPGenerator.java Trail.java
```

(to run the program)
```
java -cp .;trail_lib/* -Djava.library.path=trail_native/win64
TrailLPGenerator
```

On Mac:

(to compile the files)
```
javac -cp .:trail_lib/* TrailLPGenerator.java Trail.java
```

(to run the program)

```
java -cp .:trail_lib/* -Djava.library.path=trail_native/mac TrailLPGenerator
```

On Linux (64 bit):

(to compile the files)

```
javac -cp .:trail_lib/* TrailLPGenerator.java Trail.java
```

(to run the program)

```
java -cp .:trail_lib/* -Djava.library.path=trail_native/ux64
TrailLPGenerator
```

`TrailLPGenerator` will take several options to generate an LP file, which is then fed to the `lp_solve` program within the Java program, and the output from the solver is printed.

Command line options:

```
-n <number-of-trails>: specifies the number of trails to find
-d <distance>: specifies the maximum distance, in miles, such that all trails
      must be within this distance.
--elev-min <elevation>: specifies the minimum total elevation, that the sum
      of all the trails' elevation gains should meet or exceed
--elev-max <elevation>: specified the maximum total elevation, that the sum
      of all the trails' elevation gains should not exceed
--single-elev-min <elevation>: specifies the minimum elevation (in feet) that
      no trail's elevation gain fall below
--single-elev-max <elevation>: specifies the maximum elevation (in feet) that
      no trail's elevation gain should exceed
--single-length-max <length>: specifies the maximum length (in round trip
      miles) that no trail's length should exceed
--min-num-ratings <ratings>: the minimum number of ratings a trail should
      have to be considered in a solution
--start-lat <latitude>: the latitude of the starting location (by default Red
      Square)
--start-lng <longitude>: the longitude of the starting location (by default
      Red Square)
--min-trail-dist <distance>: the minimum distance (in miles), such that no
      two trails are closer than this distance to each otehr
--obj <obj_type>: the type of objective function to use:
      0: trails get a score of that trail's number of ratings
      1: trails get a score of that trail's rating (0.0-5.0)
      2: trails get a score of that trail's rating, raised to the power of e
(2.712), multiplied by the trail's number of ratings. E.g:

      exp(rating) * num_ratings

      3: trails get a score of that trail's rating squared
```

Examples

```
java -cp .;trail_lib/* -Djava.library.path=trail_native/win64
TrailLPGenerator -n 5 -d 25 --single-length-max 20 --single-elev-max 2000 --
elev-min 2500 --min-num-ratings 5 --obj 0
```

The above command searches for 5 trails, all within 25 miles of Red Square, with no hikes longer then 20 miles, no hikes having more than 2000 feet of elevation gain, and with all 5 trails having at least 2500 feet of elevation gain total, across all of them. In addition it selects only trails that have at least 5 ratings, and it uses the number of ratings only to score the trails (`--obj 0`)

This command is what solves the original problem for everesting:

```
java -cp .:trail_lib/* -Djava.library.path=trail_native/ux64 TrailLPGenerator
-n 5 -d 75 --single-length-max 10 --single-elev-max 8200 --elev-min 20000 --
min-num-ratings 5 --obj 1
```

Appendix C: Format of the LP file

The format of the LP file produced by the program is relatively simple, as a large amount of hikes are filtered out before the solver is used. The main variables are set as `x_i`, where each trail in the overall list of trails is assigned a numeric index for use in the solver.

Maximimum line:
Computes a score for each trail according to the objective function set (e.g. square of ratings, raw ratings), multiplies each trail variable with this score, and does the sum.

```
max: 3.8*x_0 + 3.8*x_1 + ... + 3.0*x_403 + 4.0*x_404;
```

Number of trails constraint:
This line will force the sum of the all the trail variables to be exactly 5, which will force exactly 5 hikes to be chosen:

```
x_0 + x_1 + ... + x_403 + x_404 = 5;
```

Elevation gain constraints:
These constraints force the total elevation gain to be within a range of values. Each line is of the form:

```
400.0*x_0 + 1100.0*x_1 + ... + 200.0*x_403 + 2658.0*x_404 >= 20000;
```

In this example for setting a lower bound. Setting an upper bound is identical, just using <= instead.
Binary constraints:
All variables are forced to be binary, so that line is simply:

```
bin x_0, x_1, ..., x_403, x_404
```

When the distances between trails have to be above certain values, additional constraints are added.

Distance between trails constraints:
For each unique pair of hikes, the four constraints described in the model must be listed. For example with the same constraints as the original problem, but enforcing a distance of 10 miles between each trail, these constraints are added to the LP file:

```
y_0_1 >= x_0 + x_1 - 1;
y_0_1 <= x_0;
y_0_1 <= x_1;
10000.000 - 10000.000*y_0_1 + 63.135*y_0_1 >= 10.000;
y_0_2 >= x_0 + x_2 - 1;
y_0_2 <= x_0;
y_0_2 <= x_2;
```

```
10000.000 - 10000.000*y_0_2 + 0.000*y_0_2 >= 10.000;
...
y_0_404 >= x_0 + x_404 - 1;
y_0_404 <= x_0;
y_0_404 <= x_404;
10000.000 - 10000.000*y_0_404 + 60.696*y_0_404 >= 10.000;
...
y_403_404 >= x_403 + x_404 - 1;
y_403_404 <= x_403;
y_403_404 <= x_404;
10000.000 - 10000.000*y_403_404 + 19.770*y_403_404 >= 10.000;
```

In total if there are n total hikes, then there must be $\binom{n}{2}$ groups of these constraints, which dramatically increases the size of the LP file.

Each of the y_{ij} variables must also be enforced to be binary, so in addition to the binary constraint line above, the list of y_{ij} variables will follow the variables for each hike. The overall binary constraint line will then look like:

```
bin x_0, x_1, ..., x_403, x_404, y_0_1, y_0_2, ..., y_0_404, y_1_2, ...,
y_403_y_404;
```

When a distance of 10 miles between each hike is enforced for the original problem, the LP file ends up being around 400,000 lines long. This is, however, largely due putting each variable on its own line to ensure that the 256 character line limit for the LP file format is not broken.