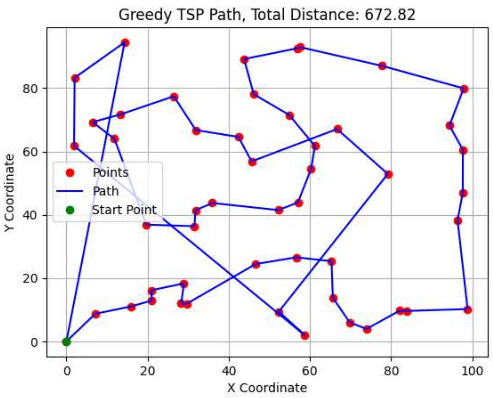
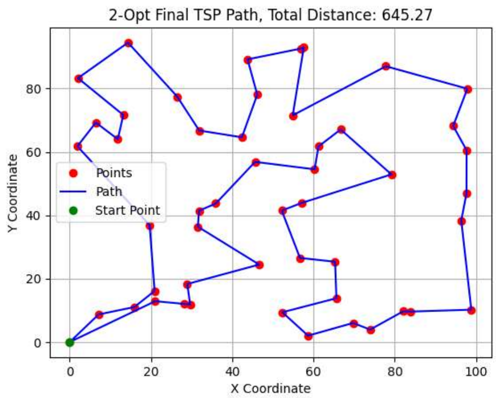
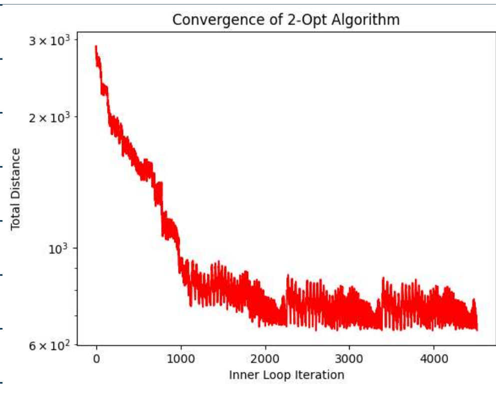
Hw 7

Wednesday, April 2, 2025 10:23 AM





Greedy: 672.82
2-Opt: 645.72
2-Opt is a very simple algorithm that is basically like a local brute force optimizer. It starts with a random route, then it iterates through each possible combination of the route indices (ie a double for loop) it then swaps the order of that set of indices and sees if it improves distance. If so, it accepts the improvement. I originally started with a genetic algorithm, but was running into some issues, so I went with this. It is heavily reliant on a good starting point, and while it runs very fast, it is not necessarily efficient, and isn't guaranteed to be better than greedy. In this case, it did improve by (672.82 - 645.27)/672.82 * 100 = 4.0947 % which is honestly not too great. I feel like this, coupled with a genetic algorithm could be quite powerful as then it wouldn't be stuck in just a local minimum.



7.2 Flying from Salt Lake City (A) to Harare, Zimbabwe (Q) You are flying from SLC (represented as A in the following table) to Harare, Zimbabwe (represented as Q) which will take several connections (various connections B through P). The cost for each of those flights is represented in the following matrix (another way to represent a directed graph with edge weights).

greedy O

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
A	-	10	7	4	8	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	6	7	3	4	-	-	-	-	-	-	-	-
C	-	-	-	-	-	9	3	10	7	-	-	-	-	-	-	-	-
D	-	-	-	-	-	9	10	5	8	-	-	-	-	-	-	-	-
E	-	-	-	-	-	10	5	3	7	-	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-	-	5	8	1	-	-	-	-	-
G	-	-	-	-	-	-	-	-	-	5	8	7	-	-	-	-	-
H	-	-	-	-	-	-	-	-	-	4	5	7	-	-	-	-	-
I	-	-	-	-	-	-	-	-	-	4	5	9	-	-	-	-	-
J	-	-	-	-	-	-	-	-	-	-	-	-	9	9	8	10	-
K	-	-	-	-	-	-	-	-	-	-	-	-	4	10	6	4	-
L	-	-	-	-	-	-	-	-	-	-	-	-	8	2	4	3	-
M	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8
N	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	7
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	7

Figure 1: Cost between locations (from the row location to the column location)

First, using a greedy algorithm, what would the cost be to fly from A to Q?

Second, apply the dynamic optimization algorithm from class to discover the optimal path (i.e. minimum cost) for flying from A to Q? Report your answer and show your work.

Greedy:

Path: A→D D→H H→J J→O O→Q

Cost: 4 + 5 + 4 + 8 + 9 = $\boxed{30}$

Dynamic Programming → See excel sheet

Optimal Path: [A, D, F, L, O, Q]

Cost: 23

Quite a large amount of work went into starting from Q and working backwards to obtain the costs all along the way. This was done by essentially using the Belleman equations.

	A	B	C
1	**Node**	**Cost**	**Next**
2	**A**	23	D
3	**B**	16	F
4	**C**	19	F or G
5	**D**	19	F
6	**E**	19	H
7	**F**	10	L
8	**G**	16	L
9	**H**	16	K or L
10	**I**	16	K
11	**J**	16	N
12	**K**	11	P
13	**L**	9	O
14	**M**	8	Q
15	**N**	7	Q
16	**O**	9	Q
17	**P**	7	Q
18	**Q**	0	Na

7.3 Branch and Bound

Solve the following problem using a branching and bounding algorithm. Show your work and demonstrate your logic as you prune or home in on the optimal solution. (You can't just brute force it. Technically, brute force is reasonable with only 5 binary variables but it'll hurt you on the final if you don't practice the method today.) The optimization for the sub-problems can be done with an off-the-shelf optimizer of your choice (such as scipy.minimize).

$$\min_{x} \quad -5.6x_1 - 7.0x_2 - 7.8x_3 - 4.0x_4 - 2.9x_5 \tag{1}$$

$$\text{subject to:} \quad 0.8x_1 + 5.9x_2 + 3.8x_3 + 1.8x_4 + 0.8x_5 \leq 8.2 \tag{2}$$

$$3.5x_1 + 2.1x_2 + 7.8x_3 + 2.2x_4 + 7.9x_5 \leq 10.2 \tag{3}$$

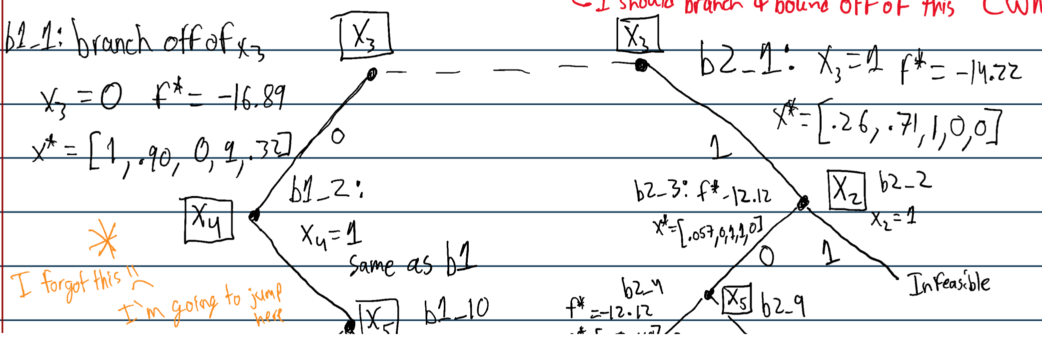
$$3.8x_1 + 2.6x_3 + 1.6x_4 \leq 8.3 \tag{4}$$

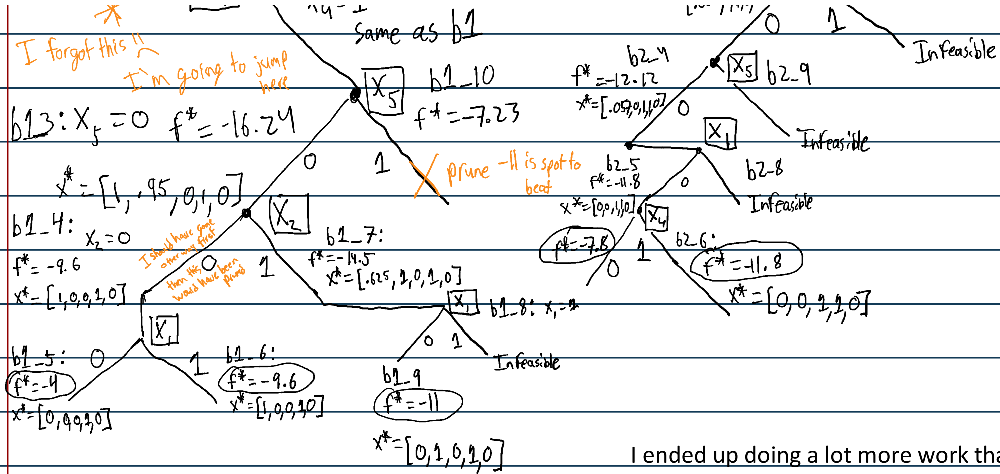
$$x_i \in \{0, 1\} \text{ for all } i \tag{5}$$

Problem 3 (branch and bound) Adding more constraints means I will do worse. The relaxed problem is a lower bound

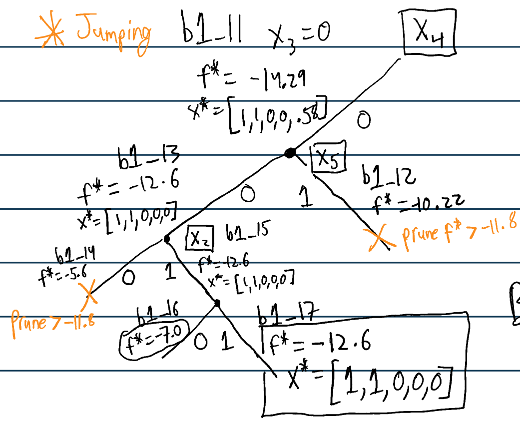
b0: $X_i = [0.5] \rightarrow x^* = [1, .699, .389, 1, \approx 0]$ & $f^* = -17.524$

← I should branch & bound off of this (whatever is furthest away from 0 or 1 (or closest to 0.5))





I ended up doing a lot more work than I needed to, as I forgot to check that first x4 branch, and I went in the wrong order as well which gave me a lot more work. In the future I will make sure to bound the values closest to 0.5 first.



Best Point = $\boxed{x^* = [1,1,0,0,0] \quad f^* = -12.6}$