Kyle Hippe 300518740

COMP 361 Design and Analysis of Algorithms

Andrew Lensen

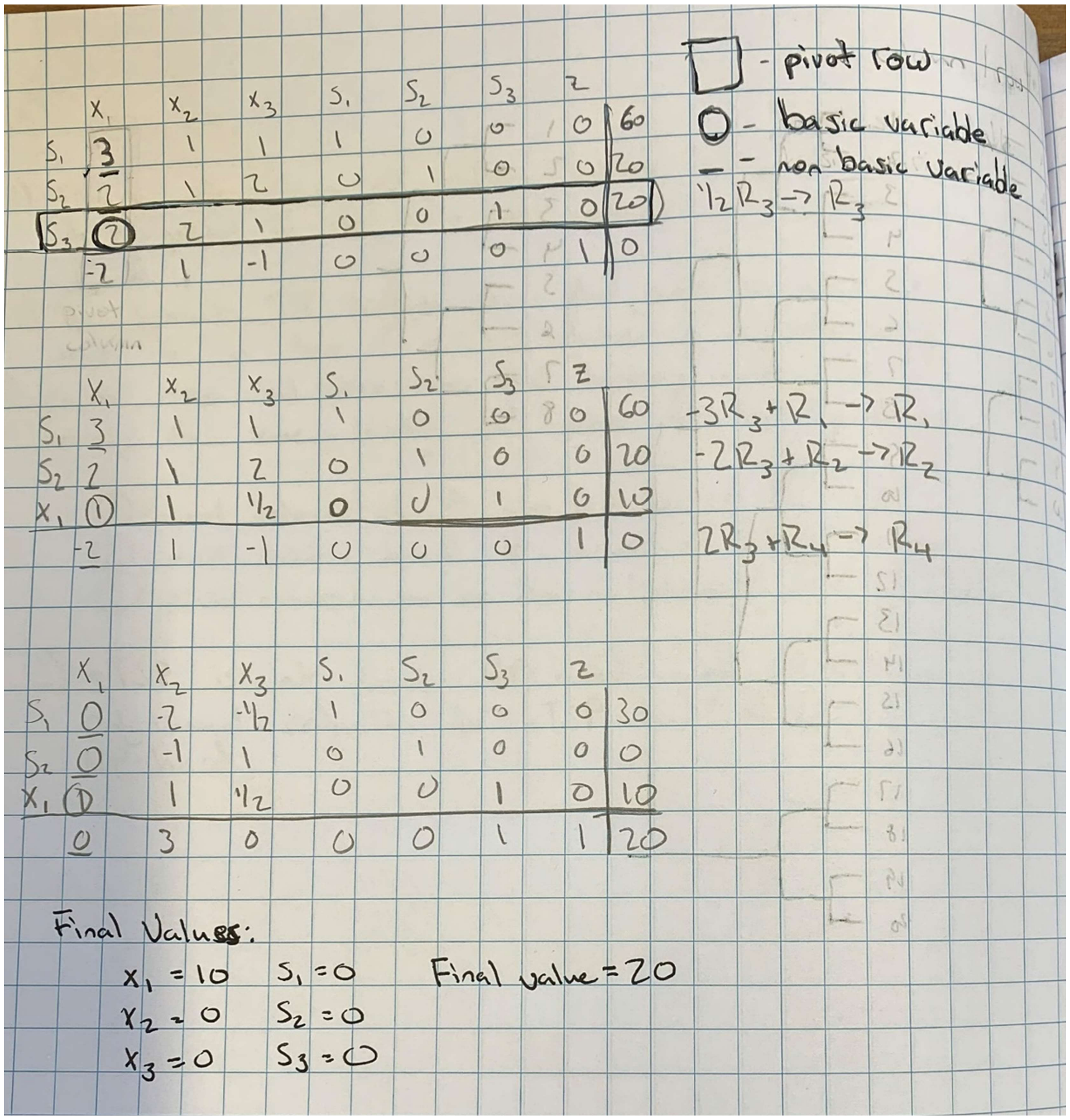
19 August 2019

Assignment 4

Simplex Algorithm

1.

The picture below shows the solution to the linear programming problem:



☐ - pivot row

○ - basic variable

— - non basic variable

	x_1	x_2	x_3	s_1	s_2	s_3	z	
s_1	3	1	1	1	0	0	0	60
s_2	2	1	2	0	1	0	0	20
s_3	2	2	1	0	0	1	0	20
	-2	1	-1	0	0	0	1	0

pivot column

$\frac{1}{2}R_3 \rightarrow R_3$

	x_1	x_2	x_3	s_1	s_2	s_3	z	
s_1	3	1	1	1	0	0	0	60
s_2	2	1	2	0	1	0	0	20
x_1	1	1	½	0	0	1	0	10
	-2	1	-1	0	0	0	1	0

$-3R_3 + R_1 \rightarrow R_1$

$-2R_3 + R_2 \rightarrow R_2$

$2R_3 + R_4 \rightarrow R_4$

	x_1	x_2	x_3	s_1	s_2	s_3	z	
s_1	0	-2	-½	1	0	0	0	30
s_2	0	-1	1	0	1	0	0	0
x_1	1	1	½	0	0	1	0	10
	0	3	0	0	0	1	1	20

Final Values:

$x_1 = 10$ $s_1 = 0$ Final value = 20

$x_2 = 0$ $s_2 = 0$

$x_3 = 0$ $s_3 = 0$

2. The easiest way to modify this problem so that the simplex algorithm could no longer be used would be to change all the constraints so that there is no upper bound. This would mean that

there would only be a lower bound, and the function could increase to infinity. This would make the constraints look like this:

s.t. $3x1 + x2 + x3 \geq 60$,

$2x1 + x2 + 2x3 \geq 20$,

$2x1 + 2x2 + x3 \geq 20$,

$x1, x2, x3 \geq 0$.

This change would make the simplex algorithm no longer work for this set of constraints.

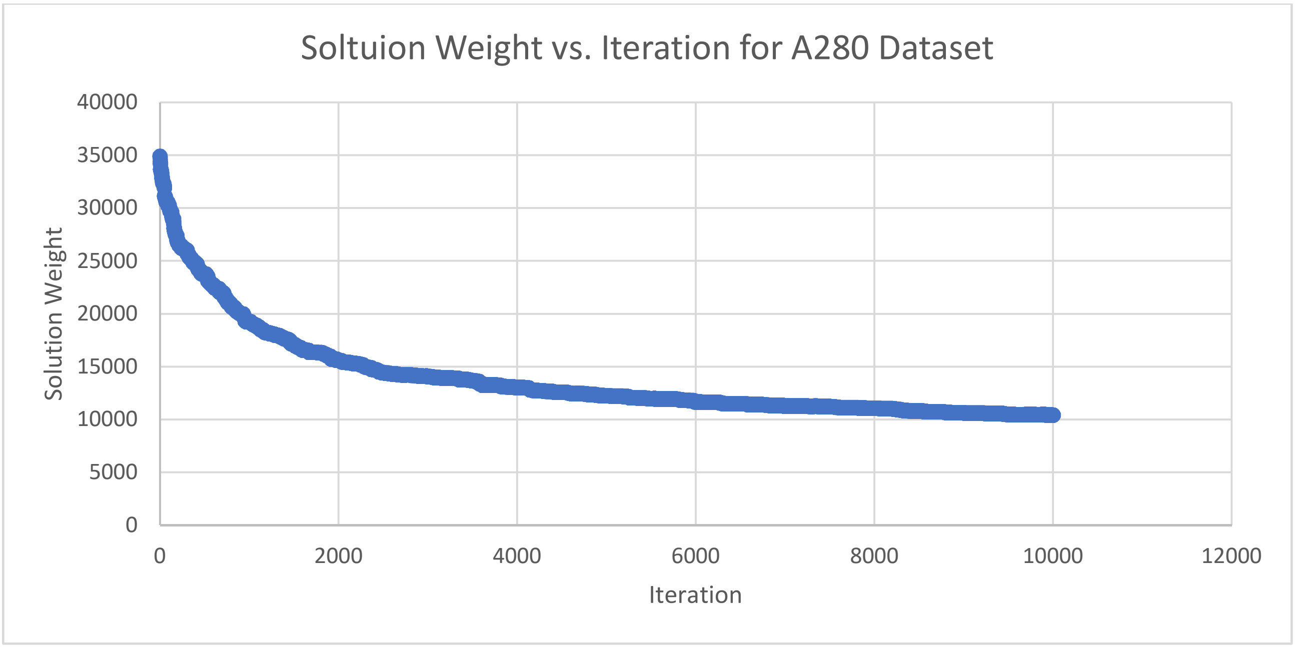
Approximation

The approximation algorithm uses a minimum spanning tree heuristic to approximate the best path to travel given a graph of vertices, where the start and end vertices must be the same. Testing this heuristic on the 'a280.tsp' dataset, we get a total solution weight of 3551.1 compared to the optimal solution of 2579. This means that this solution is 1.37 times worse than the optimal solution. This is better than the maximal 2 times worse that can be assumed from a worse case solution, and is achieved in polynomial time, not O(n!) time.

Simulated Annealing

The simulated annealing approach uses the idea of annealing (the process of cooling metals slowly to produce a more optimal structure) to randomly move through solutions, and if a new solution is not better than the previous solution, then there is a chance, based on the difference between weight of the new solution and old solution divided by the temperature, that we will accept the worse solution in hopes of reaching a more optimal solution later. The first solution was generated randomly from the list of all the nodes rather than using an MST or other heuristic, so that would be an area for improvement were I to do this portion of the assignment again. My neighborhood consisted of two nodes in the list of nodes. These two nodes were randomly selected on each iteration and were swapped to create the new solution. This hueristic was used for 10,000 iterations. It should be noted as well that the constants used for this test were as follows, the temperature was set to 25, alpha was set to 1, and a logarithmic cooling schedule was used.

The graph representing solution weight on iteration is shown below:



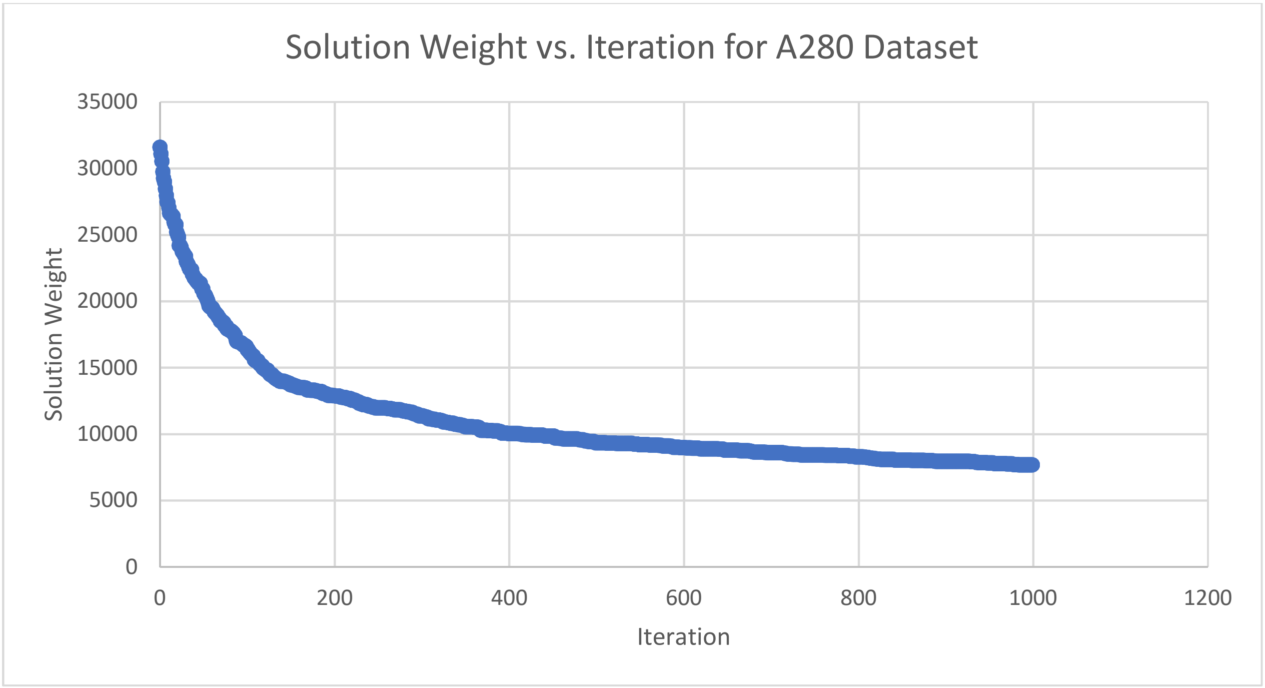
As we can see, the data shows an inverse logarithmic function, and our final solution (even though it is hard to tell from the graph) is 10,419. This solution is 4.03 times the optimal solution, but given it only went through 10,000 iterations, this is a very great improvement over the 280! possible combinations for this specific data set. This function does not give as close of a solution as the approximation algorithm does, however, it is able to run through possible combinations at a very fast rate, and given enough time, would approach the optimal solution, possibly creating a better solution than the approximation algorithm, however my implementation did not represent this. This algorithm would be more appropriately used after an approximation algorithm had established a possible solution, and more testing would be needed to be done to conclude any benefits from that approach. So, as it stands the simulated annealing algorithm when tested under the above constraints, generates a much worse solution for this data set than the MST heuristic does, most likely due to the randomness of solution generation and alteration throughout the runtime.

Evolutionary

The evolutionary approach to this algorithm is to take a population of solutions, rank them from most fit (least amount of total distance) to least fit (most amount of weight) and then take individuals from the population and, using a selection method, choose individuals and to 'breed' them. That is, take portions of each parents solution and graft them together to form a new individual with a unique solution from each parent. This process is done until a new population of solutions is established, and the process repeats.

My specifications for this problem are as follows. Each generation has 100 individuals and the first population of individuals consists of induvial solutions randomly generated from a list of nodes in the graph. Each solution is then ranked by lowest weight, and then the top 10 individuals (10% of the population) are automatically included in the next generation, this is my elite population, and they are used to keep the best solutions, making sure that there will always be good solutions in the population. The remaining induvials are created using the breeding process. The breeding process uses a tournament selection from 8 randomly selected individuals from the current population, and then the order crossover method is used to take a swath of consecutive 'genes' from parent one, and the remaining 'genes' from parent two, and combining them into a unique child. Each individual has a 1% chance to mutate. The mutation process randomly selects from either just switching two nodes in the solution (similar to the simulated annealing neighborhood) and reversing a randomly selected swath of nodes in the solution, each with a 50% chance of being selected. This process is then repeated for the remaining 90 individuals, and the process is run for 1000 generations.

The graph below shows the results of this approach to the evolutionary algorithm does for the 'a280' data set.



As we can see, the data roughly follows an inverse logarithmic curve, which is to be expected from the algorithm.

The final score reached was 7673.1, which is 2.93 times worse than the optimal solution of 2579, placing the performance of this algorithm as worse than the approximation algorithm and better than the simulated annealing algorithm. This can be accounted for by the iteration (and time) constraints of this algorithm as the test was only run with 1000 iterations. Given more time and resources, this algorithm would have eventually reached an optimal, or nearly optimal solution. Another notable difference is the randomness of the evolutionary approach and simulated annealing approach versus the MST heuristic. The randomness of these two approaches means that each iteration could either take the solution closer to an optimal solution, or farther away from the optimal solution. This means that while the evolutionary approach and the simulated annealing algorithm could get to perfectly optimal solutions, this would take lots of time and iterations on average. Comparatively, the MST algorithm would most likely never choose a perfectly optimal solution, especially for large datasets, but will consistently provide a no worse than 2 times worse solution, for every data set. This highlights the requirements for finding appropriate algorithms for the problem, as they each come with drawback, the MST heuristic never finding a perfectly optimal solution, and the simulated annealing and evolutionary approaches taking more time and resources to find an optimal solution

In order to improve the results of this algorithm, different mutation rates, generation sizes, and iterations would have to be altered in order to find the optimal set up for solving this particular instance.

Complexity

1.If an optimal solution to a TSP problem has a length L, then given a walk W we can check if this solution is optimal by simply visiting the first node, calculating the distance to the next node, adding this weight to a cumulative sum, and then moving to the next node, calculating the distance from that node to its next node and so on until we have a sum weight for W. If this cumulative weight is equal to L, then walk W is the optimal solution to the TSP problem, and if it not equal, the W is not the optimal solution

2. The TSP problem states that given an undirected graph G, a tour T is a solution if it visits every vertex in the graph once and returns to the original vertex in the minimum distance possible.

The Hamiltonian cycle problem states that given an undirected graph G, a cycle C is a solution if it visits every vertex in the graph once and returns to the original vertex.

Any instance of a Hamiltonian cycle problems can be immediately converted into instances of TSP problems, by merely changing the criteria to be a minimization. Also, any solution to a TSP problem can be converted into a solution for a Hamiltonian Cycle problem as any TSP solution visits every node in the graph and returns to the original starting position, as stated by the requirements of the Hamiltonian cycle problem. Since there is no conversion necessary, this can be done in at least polynomial time, proving that the Hamiltonian cycle problem is polynomial reducible to the TSP problem.

3. Given that the Hamiltonian Cycle problem is polynomial reducible to the TSP problem, and the TSP problem is an element of the NP class, we know that the TSP problem is at least NP-Hard, since the Hamiltonian cycle problem is in the class NP-Hard and the Hamiltonian cycle problem is polynomial redactable to the TSP problem.