TSP Report Assignment 1

Submitted by: Phalguni Rathod | Student ID: R00183770 | Course: MSc AI

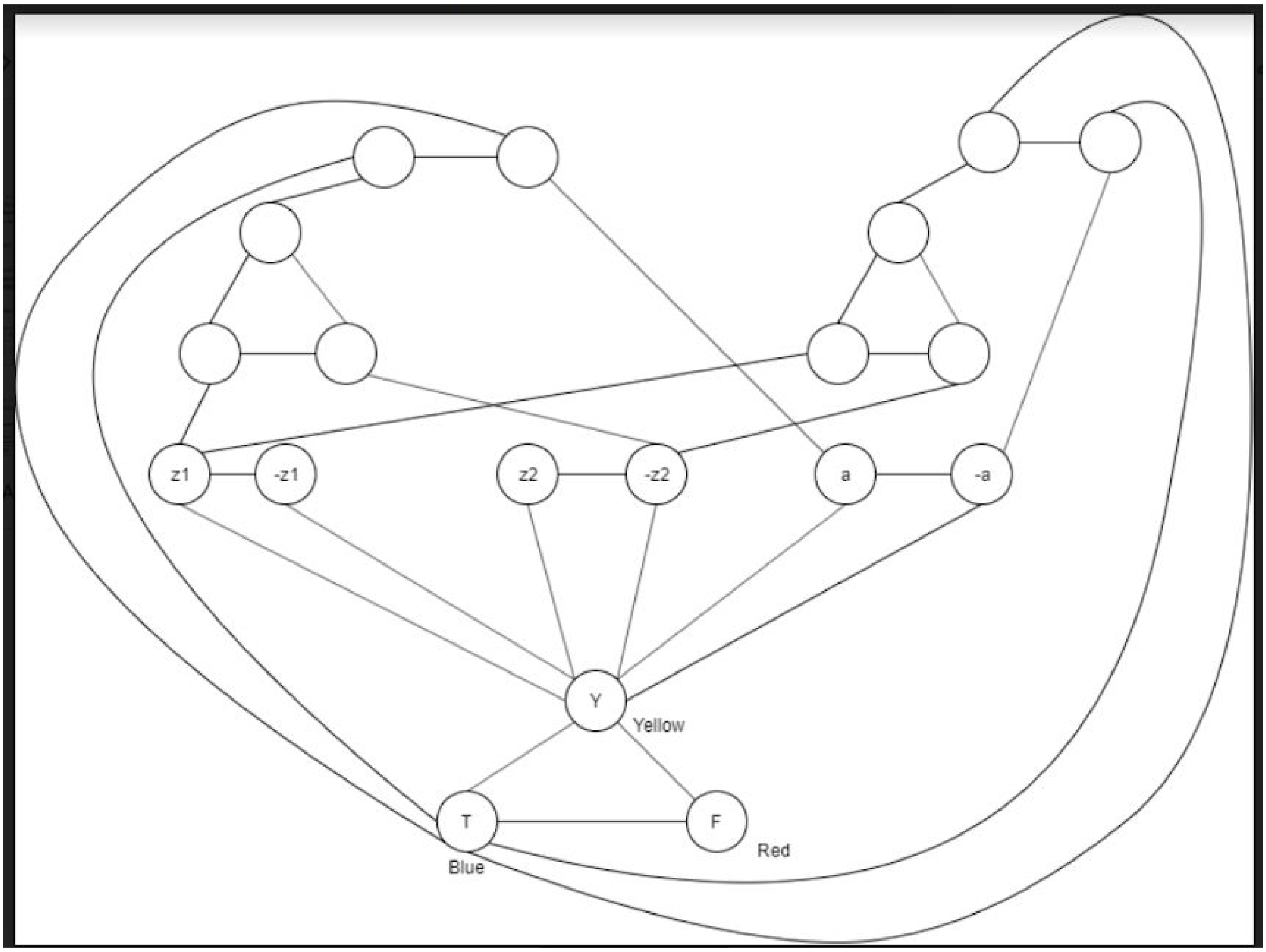
Part 1

Student Id: R0018377**0,** Using **F = (z1 ⋁ -z2) ⋀ (-z1 ⋁ z2 ⋁z3 ⋁z4 ⋁ z5)**
Solving above equation using 3SAT
F' = (Z1 ⋁ -Z2 ⋁ a) ⋀ (Z1⋁ -Z2 ⋁ -a) ⋀ (-Z1 ⋁ Z2 ⋁ b) ⋀ (-b ⋁ Z3 ⋁ c) ⋀ (-c ⋁ Z4 ⋁ Z5)

First Name: **P**halguni. Hence using **first & second subclause:**

- ○ **(Z1⋁ -Z2 ⋁ a) ⋀ (Z1 ⋁ -Z2 ⋁ -a)**

Z1 = True, -Z2= False, a = True, -a = False



Part 2: TSP Using Genetic Algorithm

We are trying to find the nearest to optimal path for TSP (Travelling Salesman Problem) using Genetic Algorithm concept.
We divided this problem into 4 Main parts

- Initialization of population (creating multiple paths for given set of cities)
- Selection of parents from population (or cities)
- Crossover: Mating the parents to create a child
- Mutation: Making minute changes in the child to make it more diverse and reduce the chances the of having same childs
- Update the population with newly created child

Initialization:

In our approach we are using the following initialization techniques:

- Random Initialization:
    - It randomly picks any 2 cities (genes) your input and puts it in a list (individuals/chromosomes).
    - This is done repetitively until all cities are part of this list
    - Such individuals/chromosomes/lists are repetitively created till required population size(default value given in our code) is met.
- Heuristic Initialization:
    - This method will choose the first gene randomly & then looks for the gene nearest to it and adds it to the list.
    - Also, done till all genes are part of chromosomes.
    - This process is repeated until the required population size is met.

Selection:

Here we have to select 2 chromosomes from the population for mating. We are using the following logic:

- Random Selection:
    - It is purely random way to select any 2 chromosomes from given population
    - We simply use random.randint() to take a list/chromosome from population
    - And return it for mating/crossover
- Stochastic Universal Selection:
    - Fitness:
        - It is the measure of quality of the chromosome. The measure is defined by the coder depending on the use case. For TSP the measure is “Path Cost”. And as we want it to be minimal, TSP is called minimization problem.
        - Path cost for us is the euclidean distance between given cities
    - Minimization:
        - There are many techniques to minimize problem but here we are simply inversing the fitness.
        - Drawback of inversing is if fitness is 0, then 1/0 will fail
        - But in TSP, fitness will never be zero as euclidean distance won’t be zero between 2 cities, Hence it is safe to use it.
    - Normalization:
        - To bring the minimized fitness to a same scale we normalize it by dividing it with sum of minimized fitness
    - Selection:
        - Now, we have to select chromosome which are the fittest ones.
        - Multiple selection of fittest chromosome is the motive of SUS
        - So, the child can also be a fit one.

        - For this we create a ruler of fitness value of all chromosome adding one after other.
        - Then we equally divide the ruler with P=fitness/N (N can be anything, but have take it as no_of_parents)
        - Make a pointer randomly between (0,P) and start selecting the chromosome & puting in fit parent list
        - Keep adding P to it and keep selecting until N is reached.
        - Randomly select and return 2 chromosome from fit parent list.

Crossover:

Now, we combine the genes of 2 selected parents from above and we are using:

- PMX Crossover:
    - Here we take certain portion/slice of P1 and put it at same position in C2 and similarly with P2 & C1.
    - Then we check existence of element from P1 in C1
    - if exist then goto it and check the element at same position in C2, if this element doesn’t exist then put it in C1.
    - Else, repeat the above step until an element is found to be put in C1
    - If element doesn’t exist in C1 then directly put it in C1.
    - Same for P2 & C2
    - And return the newly created child(ren)
- Uniform Crossover:
    - We randomly fix the position of elements in P1 & P2 and put them at the same position in C1 & C2
    - Then Compare each element from P2 and check if exist in C1, if exist then move to next element of P2.
    - If doesn’t exist then put element in C1 and move to next vacant position of C1 and next element of P2.
    - Same for P1 & C2
    - And return the newly created child(ren)

Mutation:

We make minute changes in the child to make it more diverse & reduce the possibility of same individual in the population.

- Reciprocal Exchange Mutation:
    - Simplest of all, we have to randomly choose genes/cities from the child and swap them.
    - Creates very small change.
- Inversion Mutation:
    - We simply pick/slice a portion of chromosome and reverse it and put it back to it position.
    - Here the slice is randomly taken with no fixed size of slice.

New Generation:

- The newly created child is then added back to population.
- This process is called replacement.
- And are doing full replacement by putting the child back in the population by overriding other chromosome.

Last Name : Rathod

Hence, using file inst-4, inst-16, inst-6

Instance - 4 File.

File description:
The file have **190** cities.

Config 1: Random Selection, Random Initialization, Uniform Crossover, Inversion Mutation

Config	Run	Time (mins)	Best Solution	Population size	Mutation Rate	Total Iterations
1	1	2.61	20835251.22981754	100	0.1	500
1	2	2.54	20835251.23	100	0.1	500
1	3	3.579	20835251.2298	100	0.1	700
1	4	3.78	20835251.23	100	0.1	650
1	5	5.25	20835251.2299	100	0.1	900

Experiments:

- Keeping Population Size & Mutation rate constant as 100 & 0.1 respectively
- Varying no. of iterations

Observations:

- On increasing the number of iterations, still there is no major change in the Best Solution.
- Increase in # of iterations increases the time to run the whole program proportionately.

Numeric highlights:

- **Best Solution:** 22981754
- **Time Taken: 2.6 min**
- **Mean Best Solution: 20835251.229904**
- **Median: 20835251.2299**

Config 2 : Random Selection, Random Initialization, PMX Crossover, Reciprocal Exchange Mutation

Config 2	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total Iter
2	1	1.32	21924290.9850463	20476550.3635069	405	100	0.1	500
2	2	1.59	21604957.9305457	20793023.2534658	476	100	0.1	600
2	3	1.95	21829453.1786015	20290896.3235976	359	100	0.1	700
2	4	2.22	21531466.6365140	20357095.5320097	750	100	0.1	800
2	5	2.24	22280552.5406247	20636671.4785208	634	100	0.1	900

Experiments:

- Keeping the Population & Mutation Rate Constant as given (100, 0.1)
- Varying no. of iterations, incrementing it by 100

Observations:

- Significant Variation in best solution when we vary the total iterations
- The time taken is gradually increasing with increase in total iterations

Numeric Highlights:

- Best Solution: 20290896.323
- Iter: 359
- Time taken: 1.95 mins
- Total Iter: 700
- Mean: 20510847.39
- Median: 20290896.3235

Config 3: Stochastic Universal Sampling, Random Initialization, Uniform Crossover, Reciprocal Exchange Mutation

Config 3	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total Iter
3	1	12.85	21697919.51	20565444.99	63	300	0.1	500
3	2	14.90	22245344.61	20387624.78	224	300	0.1	600
3	3	16.01	21751406.377	20706342.4478	430	300	0.1	700
3	4	13.56	21449424.57	20545904.5412	290	400	0.1	500
3	5	18.528	21923687.7291	20711850.358	215	400	0.1	600

Experiments:

- Varying the no. of iterations
- Varying the number of population

Observation:

- There is a gradual increase in time when iterations are increased
- But significant time is increased by increasing the population.
- One reason could be the increased number of population lead to more individuals to be created. Hence, Random Initialization is taking time.

Numeric Highlights:

- **Best Solution**: 20387624.78
- **Iter:** 224
- **Time Taken**: 14.90 mins
- **Mean Best Solution:** 20583433
- **Median**: 20706342.447

Config 4: Stochastic Universal Sampling, Random Initialization, PMX Crossover, Reciprocal Exchange Mutation

Config 4	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total iter
4	1	1.70	21924290.9850	20846688.7667	244	100	0.1	500
4	2	2.05	21423306.4323	20439293.6584	102	100	0.1	600
4	3	2.31	21661660.4480	20839070.9708	242	100	0.1	700
4	4	1.62	22105472.7487	20900366.9136	400	100	0.5	500
4	5	1.99	22240784.8904	21067044.8727	81	100	0.5	600

Experiments:

- Keeping the population constant
- Varying the Total Iterations incrementing by 100
- Varying the mutation Rate

Observation

- Nominal Increase in Time with increase in iteration
- With increase in mutation rate, it was expected that more diversification can happen & the Best solution may be better but couldn't observe it

Numeric Highlight:

- **Best Solution:** 20439293.6584
- **Iter:** 102
- **Time Taken:** 2.054 mins
- **Mean Best Solution:** 20818493
- **Median:** 20846688.766

Config 5: Stochastic Universal Sampling, Random Initialization, PMX Crossover, Inversion Mutation

Config 5	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation	Total iter
5	1	1.654	21924290.9850	20904190.14070	142	100	0.1	500
5	2	2.173	21860352.6831	21034998.80517	67	100	0.1	600
5	3	2.395	21533393.5750	20672325.52400	331	100	0.1	700
5	4	4.970	22033416.3791	20494709.09785	453	200	0.5	500
5	5	6.375	21251577.8440	20469488.89062	144	200	0.5	600
5	6	7.054	21554039.7000	20567818.69502	104	200	0.5	700

Experiments:

- Varying the Iteration by incrementing by 100 in combination with Mutation Rate & Population
- Keeping Mutation Rate as 0.1 & 0.5
- Changing population as 100 & 200

Observations:

- Significant increase in time if we increase the population & mutation rate, approx 2 min rise per iteration
- With increase in mutation rate, we can expect more diverse population & better solution.
- And above expectation has come true

Numeric Highlights:

- Best Solution: 20469488.89062
- Iter: 144
- Time Taken: 6.375 mins
- Mean Best Solution: 20690588.53

- Median: 20620072.1095

Config 6: Stochastic Universal Sampling, Random Initialization, Uniform Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation	Total iter
6	1	1.847	21924290.9850	20734806.9265	340	100	0.1	500
6	2	2.326	21969085.0673	20794423.8996	142	100	0.1	600
6	3	2.602	21633589.3712	20920381.0338	240	100	0.1	700
6	4	4.903	21536450.4417	20525460.0097	23	200	0.5	500
6	5	6.109	21708716.4779	20700186.8862	421	200	0.5	600
6	6	7.110	22151674.1342	20822827.2074	429	200	0.5	700

Experiment:

- Varying the Iteration by incrementing by 100 in combination with Mutation Rate & Population
- Keeping Mutation Rate as 0.1 & 0.5
- Changing population as 100 & 200

Observation:

- Not much increase in time with increase in no. of iterations, approx 30-40 secs
- Significant increase in time if we increase the population & mutation rate, approx 2 min rise per iteration
- With increase in mutation rate, it was expected that more diversification can happen & the Best solution may be better but we got 2 conflicting best solution, highest & lowest (Run 4 & 6)
- This can be due to 0.5 Mutation Rate as it only allows 50% chances of mutation. SO, one of the Run must have not been mutated

Numeric Highlights

- **Best Solution:** 20734806.9265
- **Iter:** 23
- **Time Taken:** 4.903
- **Mean Best Solution:** 20749680.993867
- **Median: 20764615.4131**

Config 7: Stochastic Universal Sampling, Heuristic Initialization, PMX Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Population	Mutation	Total iter
7	1	18.90	3933557.7679	3933557.7679	50	0.1	500
7	2	18.74	3885429.2345	3885429.2345	50	0.1	500
7	3	18.49	3937243.6286	3937243.6286	50	0.1	500
7	4	37.54	3885429.2345	3885429.2345	100	0.5	500
7	5	37.51	3908620.3106	3908620.3106	100	0.5	500
7	6	37.64	3885429.2345	3885429.2345	100	0.5	500

Experiment:

- Varying the combination of Mutation Rate & Population
- Keeping Mutation Rate as 0.1 & 0.5
- Changing population as 50 & 100

Observation:

- Run 2, 4, 6 are giving totally the same initial solution, best solution despite having different population size & mutation rate
- Time taken is doubled up by increasing the population & mutation
- With increase in mutation rate, it was expected that more diversification can happen & the Best solution may be better but we got 2 conflicting best solution, highest & lowest (Run 5 & 4) both in increased mutation rate.
- This can be due to 0.5 Mutation Rate as it only allows 50% chances of mutation. SO, one of the Run must have not been mutated

Numeric Highlights

- **Best Solution:** 3885429.2345
- **Time Taken: 18.49**
- **Mean Best Solution: 3905951.568**
- **Median: 3897024.772**

Config 8: Stochastic Universal Sampling, Heuristic Initialization, PMX Crossover, Inversion Mutation

Config	Run	Time	Initial Sol	Best Sol	Population	Mutation	Total iter
8	1	19.17	3933557.7679	3933557.7679	50	0.1	500
8	2	18.87	3937243.6286	3937243.6286	50	0.1	500
8	3	18.93	3885429.2345	3885429.2345	50	0.1	500
8	4	38.27	3885429.2345	3885429.2345	100	0.5	500
8	5	38.46	3900881.2900	3897213.5883	100	0.5	500
8	6	38.23	3900881.2900	3900881.2900	100	0.5	500

Experiment:

- Varying the combination of Mutation Rate & Population
- Keeping Mutation Rate as 0.1 & 0.5
- Changing population as 50 & 100

Observation:

- Run 2, 3 are giving totally the same initial solution, best solution despite having different population size & mutation rate
- Time taken is doubled up by increasing the population & mutation

Numeric Highlights

- **Best Solution:** 3885429.2345
- **Time Taken: 18.93 min**
- **Mean Best Solution: 3906625.790**
- **Median: 3899047.43**

Instance - 16 File

Config 1: Random Selection, Random Initialization, Uniform Crossover, Inversion Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation	Total iter
1	1	2.77	113216101.9938	104261219.1426	496	100	0.1	500
1	2	3.29	112334205.3630	103079377.9329	87	100	0.1	600
1	3	3.84	110817679.0356	103638200.8923	593	100	0.1	700
1	4	2.74	113915861.6463	103050637.3836	239	100	0.1	500
1	5	3.30	109929055.8972	101757748.3274	402	100	0.1	600
1	6	3.84	113073734.8116	101595695.3381	580	100	0.1	700

Experiment:

- Keeping Population & Mutation Rate as constant (100 & 0.1) as given in assignment
- Varying the number of iterations

Observation:

- Not much increase in time with increase in no. of iterations, approx 30-40 secs increment per Iteration
- Similar Best solution if all parameters(Population Size, Mutation Rate, Total Iteration) kept same over multiple runs

Numeric Highlights:

- **Best Solution:** 101595695.3381
- **Iter: 580**
- **Time Taken: 3.84 min**
- **Mean Best Solution: 102897146.50282**
- **Median: 103065007.658**

Config 2 : Random Selection, Random Initialization, PMX Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total iter
2	1	2.54	113216101.9938	104261219.1426	496	100	0.1	500
2	2	3.10	112334205.3630	103079377.9328	87	100	0.1	600
2	3	3.70	110817679.0356	103638200.8923	593	100	0.1	700
2	4	2.59	113915861.6462	103050637.3836	239	100	0.1	500
2	5	3.12	109929055.8977	101757748.3274	402	100	0.1	600
2	6	3.43	113073734.8119	101595695.3381	580	100	0.1	700

Experiment:

- Keeping Population & Mutation Rate as constant (100 & 0.1) as given in assignment
- Varying the number of iterations

Observation:

- Not much increase in time with increase in no. of iterations, approx 30-40 secs/iteration
- Comparable time taken if all parameters kept same over multiple runs

Numeric Highlights:

- **Best Solution:** 101595695.3381956
- **Iter:** 580
- **Time Taken:** 3.43 min
- **Mean Best Solution:** 102897146.5028
- **Median**: 103065007.6582

Config 3: Stochastic Universal Sampling, Random Initialization, Uniform Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation	Total iter
3	1	3.1908	113216101.9938	102933271.9230	124	100	0.1	500
3	2	4.501	108577781.4591	101969278.7573	509	100	0.1	600
3	3	4.988	111558464.9136	101206028.0756	509	100	0.1	700
3	4	7.780	106456364.2889	102119855.7211	83	200	0.5	500
3	5	11.16	107340684.5445	100256173.3425	72	200	0.5	600
3	6	11.23	109870493.4260	100580582.7101	659	200	0.5	700

Experiment:

- Keeping the population either 100 or 200
- Varying the Total Iterations incrementing by 100 from 500 to 700
- Varying the mutation Rate either 0.1 & 0.5
- Experimenting with few combination of population, total iterations, & mutation rate

Observation:

- No significant increase in time on increasing iteration & keeping other parameter as it is.approx 30-40 secs/iteration
- For population = 200 Mutation Rate =0.5, For increase by 100 iterations approx 4 min increase is expected
- On doubling up the population & keeping mutation rate of 0.5 & keeping iteration constant, time is also doubling up

Numeric Highlights:

- **Best Solution:** 100256173.3425
- **Iter: 659**
- **Time Taken:** 11.23
- **Mean Best Solution:** 101510865.08827
- **Median:** 101587653.4165

Config 4: Stochastic Universal Sampling, Random Initialization, PMX Crossover, Reciprocal Exchange Mutation

Config	Run	Time (min)	Initial Sol	Best Sol	Iter #	Population	Mutation	Total iter
4	1	7.762	113216101.9938	102769373.3107	331	100	0.1	500
4	2	10.072	113815075.5750	102141723.9827	204	100	0.1	600
4	3	8.698	112560542.4355	102249613.6890	630	100	0.1	700
4	4	7.44	108828129.5631	99401707.97930	49	200	0.5	500
4	5	9.129	108832536.1097	101212514.1176	176	200	0.5	600
4	6	20.76	110949776.7179	101049624.1467	68	200	0.5	700

Experiment:

- Keeping the population either 100 or 200
- Varying the Total Iterations incrementing by 100
- Varying the mutation Rate either 0.1 & 0.5
- Experimenting with few combination of population, total iterations, & mutation rate

Observation:

- Time is increasing on increasing iteration & keeping other parameter as it is.approx 2-3min/iteration
- For population = 200 Mutation Rate =0.5, For increase by 100 iterations approx 4 min increase is expected

Numeric Highlights:

- **Best Solution:** 99401707.97930

- **Iter:** 49
- **Time Taken:** 7.44 min
- **Mean Best Solution:** 101470759.53767
- **Median:** 101677119.0502

Config 5: Stochastic Universal Sampling, Random Initialization, PMX Crossover, Inversion Mutation

Config	Run	Time (min)	Initial Sol	Best Sol	Iter #	Population	Mutation	Total iter
5	1	3.069	113216101.9938	103035181.7489	415	100	0.1	500
5	2	3.545	113815075.5750	102090719.3718	397	100	0.1	600
5	3	4.168	112435743.4631	99941457.86569	317	100	0.1	700
5	4	7.884	111464917.2644	98644256.33507	383	200	0.5	500
5	5	9.111	111239817.6618	102727893.0219	386	200	0.5	600
5	6	10.33	110313662.6219	101650612.0361	392	200	0.5	700

Experiment:

- Keeping the population either 100 or 200
- Varying the Total Iterations incrementing by 100
- Varying the mutation Rate either 0.1 & 0.5
- Experimenting with few combination of population, total iterations, & mutation rate

Observation:

- No significant increase in time on increasing iteration & keeping other parameter (For population = 100 Mutation Rate =0.1) as it is.approx 30-40 secs/iteration
- For population = 200 Mutation Rate =0.5, For increase by 100 iterations approx 2 min increase is expected
- On doubling up the population to 200 & keeping mutation rate of 0.5 & keeping iteration constant, time is also doubling up

Numeric Highlights:

- **Best Solution:** 98644256.33507
- **Iter: 383**
- **Time Taken: 7.9 mins**
- **Mean Best Solution: 101348353.39658**
- **Median: 101870665.704**

Config 6: Stochastic Universal Sampling, Random Initialization, Uniform Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation	Total iter
6	1	3.28	113216101.9938	103711195.1681	395	100	0.1	500
6	2	3.71	109592354.6349	103108995.2655	595	100	0.1	600
6	3	4.30	109851181.9854	100710596.7559	99	100	0.1	700
6	4	7.88	108299639.9113	103624734.1460	32	200	0.5	500
6	5	12.33	107340684.5445	100043994.8895	10	200	0.5	600
6	6	31.75	111904477.2440	100179899.0903	86	200	0.5	700

Experiment:

- Keeping the population either 100 or 200
- Varying the Total Iterations incrementing by 100
- Varying the mutation Rate either 0.1 & 0.5
- Experimenting with few combination of population, total iterations, & mutation rate

Observation:

- No significant increase in time on increasing iteration & keeping other parameter (For population = 100 Mutation Rate =0.1) as it is.approx 30-40 secs/iteration
- For population = 200 Mutation Rate =0.5, For increase by 100 iterations it is increasing exponentially (specially, run: 4, 5, 6)
- On doubling up the population to 200 & keeping mutation rate of 0.5 & keeping iteration constant, time is increasing exponentially

Numeric Highlights:

- **Best Solution:** 100043994.8895
- **Iter: 10**
- **Time Taken: 12.33 mins**
- **Mean Best Solution: 101896569.21922**
- **Median: 101909796.0107**

Config 7: Stochastic Universal Sampling, Heuristic Initialization, PMX Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Population	Mutation	Total iter
7	1	46.39	7401694.9176	7401694.9176	50	0.1	500
7	2	44.85	7410032.0828	7410032.0828	50	0.1	500
7	3	42.95	7398331.2073	7398331.2073	50	0.1	500
7	4	88.31	7398331.2073	7398331.2073	100	0.5	500
7	5	86.25	7398331.2073	7398331.2073	100	0.5	500

Experiment:

- Varying the combination of Mutation Rate & Population
- Keeping iterations constant at 500
- Keeping Mutation Rate as 0.1 & 0.5
- Changing population as 50 & 100

Observation:

- Run 3,4,5 are giving totally the same initial solution, best solution despite having different population size & mutation rate
- Time taken is doubled up by increasing the population to 100 & mutation to 0.5

Numeric Highlights

- **Best Solution:**7398331.2073
- **Time Taken:** 42.95
- **Mean Best Solution:7401344.12446**
- **Median: 7398331.2073**

Config 8: Stochastic Universal Sampling, Heuristic Initialization, PMX Crossover, Inversion Mutation

Config	Run	Time	Initial Sol	Best Sol	Population	Mutation	Total iter
8	1	45.53	7401694.9176	7401694.9176	50	0.1	500
8	2	44.39	7440778.9735	7440778.9735	50	0.1	500
8	3	44.45	7451115.1306	7451115.1306	50	0.1	500
8	4	88.17	7401694.9176	7401694.9176	100	0.1	500
8	5	88.66	7412647.4325	7412647.4325	100	0.1	500

Experiment:

- Varying the combination of Mutation Rate & Population
- Keeping iterations constant at 500
- Keeping Mutation Rate as 0.1 & 0.5
- Changing population as 50 & 100

Observation:

- Run 1,4 are giving totally the same initial solution, best solution despite having different population size & mutation rate
- Time taken is doubled up by increasing the population 100 & mutation to 0.5

Numeric Highlights

- **Best Solution:**7401694.9176
- **Time Taken:** 45.53
- **Mean Best Solution: 7421586.2743**
- **Median: 7412647.4345**

Inst 6 File

Config 1: Random Selection, Random Initialization, Uniform Crossover, Inversion Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total iter
1	1	16.81	347563232.0608	338885411.1369	266	100	0.1	500
1	2	20.02	350155101.5616	337491599.3162	107	100	0.1	600
1	3	23.38	346278198.9812	334320034.1494	576	100	0.1	700
1	4	20.02	340396234.2503	336417018.6972	284	100	0.1	600
1	5	23.43	349754446.7723	346453765.2427	221	100	0.1	700
1	6	16.67	350750471.8313	334860364.2779	127	100	0.1	500

Experiment:

- Keeping Population & Mutation Rate as constant (100 & 0.1) as given in assignment
- Varying the number of iterations

Observation:

- Significant increase in time with increase in no. of iterations, approx 3-4mins increment per Iteration
- Similar Best solution if all parameters(Population Size, Mutation Rate, Total Iteration) kept same over multiple runs

Numeric Highlights:

- **Best Solution:** 334320034.1494
- **Iter: 576**
- **Time Taken: 23.38 min**
- **Mean Best Solution: 338071365.47005**
- **Median: 336954309.0067**

Config 2 : Random Selection, Random Initialization, PMX Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total iter
2	1	13.52	347563232.0608	338799490.5783	92	100	0.1	500
2	2	16.103	350380325.8168	337420861.2667	499	100	0.1	600
2	3	18.86	349275356.8444	337447363.7671	90	100	0.1	700
2	4	13.41	347968770.1540	338264567.6498	57	100	0.1	500
2	5	16.10	350455322.7394	334066727.3270	155	100	0.1	600
2	6	18.77	345046585.0720	333085202.1567	497	100	0.1	700

Experiment:

- Keeping Population & Mutation Rate as constant (100 & 0.1) as given in assignment
- Varying the number of iterations

Observation:

- Increase in time with increase in no. of iterations, approx 3-4 mins/iteration

Numeric Highlights:

- **Best Solution:** 333085202.1567
- **Iter: 497**

- **Time Taken: 18.77 min**
- **Mean Best Solution:336514035.4576**
- **Median: 337434112.5169**

Config 3: Stochastic Universal Sampling, Random Initialization, Uniform Crossover, Reciprocal Exchange Mutation

Config 3	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total iter
3	1	16.48	347563232.0608	333799625.3805	401	100	0.1	500
3	2	19.82	341796976.1264	332279093.9295	365	100	0.1	600
3	3	23.21	351110325.7434	335413565.0780	156	100	0.1	700
3	4	34.82	349179676.4108	333566233.6565	22	200	0.5	500
3	5	41.80	345411547.6743	335640015.7612	418	200	0.5	600
3	6	48.68	348670830.4381	332554760.2176	36	200	0.5	700

Experiment:

- Keeping the population either 100 or 200
- Varying the Total Iterations incrementing by 100
- Varying the mutation Rate either 0.1 & 0.5
- Experimenting with few combination of population, total iterations, & mutation rate

Observation:

- Time is increasing on increasing iteration & keeping other parameter as it is. For increase by 100 iterations approx 3 min increase is expected
- On doubling up the population & keeping mutation rate of 0.5 & keeping population constant, time is also doubling up.

Numeric Highlights:

- **Best Solution:** 332279093.9295
- **Iter: 365**
- **Time Taken: 19.82 mins**
- **Mean Best Solution: 333875549.00388**
- **Median: 333682929.5185**

Config 4: Stochastic Universal Sampling, Random Initialization, PMX Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total iter
4	1	13.88	347563232.0608	332855094.8946	374	100	0.1	500
4	2	16.34	346754743.9781	335437326.4500	398	100	0.1	600
4	3	19.26	349558499.4381	331345224.1457	273	100	0.1	700
4	4	28.99	343553275.5323	332844382.2609	23	200	0.5	500
4	5	34.48	344643307.9706	334571570.4381	308	200	0.5	600

4	6	40.30	345636053.4682	334329330.0314	106	200	0.5	700

Experiment:

- Keeping the population either 100 or 200
- Varying the Total Iterations incrementing by 100
- Varying the mutation Rate either 0.1 & 0.5
- Experimenting with few combination of population, total iterations, & mutation rate

Observation:

- Time is increasing on increasing iteration & keeping other parameter as it is. For increase by 100 iterations approx 3 min increase is expected
- For population = 200 Mutation Rate =0.5, For increase by 100 iterations approx 6 min increase is expected
- On doubling up the population & keeping mutation rate of 0.5 & keeping iteration constant, time is also doubling up

Numeric Highlights:

- **Best Solution:** 331345224.1457
- **Iter: 273**
- **Time Taken: 19.26 min**
- **Mean Best Solution: 333563821.37012**
- **Median: 333592212.463**

Config 5: Stochastic Universal Sampling, Random Initialization, PMX Crossover, Inversion Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total iter
5	1	14.62	347563232.0608	328621735.8845	229	100	0.1	500
5	2	17.43	344556891.7946	335104473.9897	239	100	0.1	600
5	3	20.28	345865800.0683	331265017.7503	404	100	0.1	700
5	4	31.06	348482077.5604	336702087.2217	117	200	0.5	500
5	5	37.43	344340359.0819	335821805.7497	274	200	0.5	600
5	6	43.68	345411074.7383	337513552.1900	579	200	0.5	700

Experiment:

- Keeping the population either 100 or 200
- Varying the Total Iterations incrementing by 100
- Varying the mutation Rate either 0.1 & 0.5
- Experimenting with few combination of population, total iterations, & mutation rate

Observation:

- Time is increasing on increasing iteration & keeping other parameter as it is. For increase by 100 iterations approx 3 min increase is expected
- For population = 200 Mutation Rate =0.5, For increase by 100 iterations approx 6 min increase is expected
- On doubling up the population & keeping mutation rate of 0.5 & keeping iteration constant, time is also doubling up

Numeric Highlights:

- **Best Solution:** 328621735.8845
- **Iter: 229**
- **Time Taken: 14.62 min**
- **Mean Best Solution: 334171445.46432**
- **Median: 335463139.8697**

Config 6: Stochastic Universal Sampling, Random Initialization, Uniform Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Iter #	Population	Mutation Rate	Total iter
6	1	17.60	348482077.5604	337120525.3911	378	100	0.1	500
6	2	20.72	347744530.2384	335794220.3096	188	100	0.1	600
6	3	23.97	343056324.4264	336153721.9339	472	100	0.1	700
6	4	35.59	343454156.1079	333268693.7897	214	200	0.5	500
6	5	43.05	344784495.2421	334500185.5868	22	200	0.5	600
6	6	50.39	347931340.2065	336540183.6957	262	200	0.5	700

Experiment:

- Keeping the population either 100 or 200
- Varying the Total Iterations incrementing by 100
- Varying the mutation Rate either 0.1 & 0.5
- Experimenting with few combination of population, total iterations, & mutation rate

Observation:

- Time is increasing on increasing iteration & keeping other parameter as it is. For increase by 100 iterations approx 3 min increase is expected
- For population = 200 Mutation Rate =0.5, For increase by 100 iterations approx 8 min increase is expected
- On doubling up the population & keeping mutation rate of 0.5 & keeping iteration constant, time is also doubling up

Numeric Highlights:

- **Best Solution:**333268693.7897
- **Iter: 214**
- **Time Taken: 35.59 min**
- **Mean Best Solution: 335562921.78447**
- **Median: 335973971.1218**

Config 7: Stochastic Universal Sampling, Heuristic Initialization, PMX Crossover, Reciprocal Exchange Mutation

Config	Run	Time	Initial Sol	Best Sol	Population	Mutation Rate	Total iter
7	1	333.71 mins(~5.5 hours)	13005325.8396	13005325.8396	50	0.1	500
7	2	318.50590728 5.5 hours)	12958912.0446	12958912.0446	50	0.1	500

Trying with Random Selection with other functions as it is

Config	Run	Time	Initial Sol	Best Sol	Population	Mutation Rate	Total iter
7	1	326.16 (~5.5 hours)	13005325.8396	13005325.8396	50	0.1	500

Experiment:

- Keeping the population constant 50
- Keeping the Total iterations constant 500
- Keeping the mutation Rate of 0.1

Observation:

- This config on running with inst-6 file with 823 nodes is taking 5+ hours to complete a single iteration
- Reason behind it is yet not pin-pointed but on investigation I am estimating the reason to be the Heuristic function as it is having multiple nested loops and hence, due to increased time complexity coupled with a huge data set of 823 cities it is taking hours to run.
- As my system couldn't support such a massive processing I tried running it on Google Colab and was able to run the above 2 iterations due to time constraints by colab.
- To check if the problem was with SUS or Heuristic, I ran the program with Random Selection and yet I got similar results.

Numeric Highlights:

- **Best Solution:**12958912.044
- **Time Taken: 326.16 min (~5.5 hours)**
- **Mean Best Solution: 12982118.94**

Config 8: Stochastic Universal Sampling, Heuristic Initialization, PMX Crossover, Inversion Mutation

Config	**Run**	**Time (mins)**	**Initial Sol**	**Best Sol**	**Population**	**Mutation Rate**	**Total iter**
8	1	658.60 (~10.9 hours)	12908737.2702	12908737.2702	100	0.1	500

Config	**Run**	**Time (mins)**	**Initial Sol**	**Best Sol**	**Population**	**Mutation Rate**	**Total iter**
8	1	333.60 (~10.9 hours)	13005325.8396	13005325.8396	100	0.1	500

Experiment:

- Keeping the population constant 100
- Keeping the Total iterations constant 500
- Keeping the mutation Rate of 0.1

Observation:

- This config on running with inst-6 file with 823 nodes is taking 10+ hours to complete a single iteration
- Reason behind it is yet not pin-pointed but on investigation I am estimating the reason to be the Heuristic function as it is having multiple nested loops and hence, due to increased time complexity coupled with a huge data set of 823 cities it is taking hours to run.
- As my system couldn't support such a massive processing I tried running it on Google Colab and was able to run the above 2 iterations due to time constraints by colab.
- To check if the problem was with SUS or Heuristic, I ran the program with Random Selection and it took half time taken by SUS. I will have to further investigate the issue to pin-point the exact reason

Numeric Highlights:

- **Best Solution:**12908737.2702
- **Time Taken: 658.60 min (~10.9 hours)**
- **Mean Best Solution:** 12908737.2702