Metaheuristic Optimization

Assignment 1

By Srinivasaraghavan Seshadhri R00195470

Part 1

1) a) To Convert the formula F below into a 3SAT formula F' and to find a solution to F' and verify that this is a solution to F.

$$F = (Z_1 \vee \neg Z_2) \wedge (\neg Z_1 \vee Z_2 \vee Z_3 \vee Z_4 \vee Z_5 \vee \neg Z_6)$$

Converting the above SAT equation F into Below 3SAT equation F'

$$F' = (Z_1 \vee \neg Z_2 \vee U_1) \wedge (Z_1 \vee \neg Z_2 \vee \neg U_1) \wedge (\neg Z_1 \vee Z_2 \vee W_1) \wedge (\neg W_1 \vee Z_3 \vee W_2) \wedge (\neg W_2 \vee Z_4 \vee W_3) \wedge (\neg W_3 \vee Z_5 \vee Z_6)$$

The conversions were done using the methodologies taught in W3 Lecture, by using the appropriate reduction formulae. U_1, W_1, W_2, W_3 are the introduced variables.

Finding a Solution for the 3SAT instance of F and verifying whether it is a solution to the original SAT problem. Now let's do the same by plugging in the below values to the respective variables.

$$Z_1 = T,\ Z_2 = T,\ Z_3 = T,\ Z_4 = T,\ Z_5 = T,\ Z_6 = F,\ U_1 = F,\ W_1 = T,\ W_2 = T,\ W_3 = F$$

Plugging in the above values to the variables in the 3SAT Equation F as follows:

$$F' = (T \vee F \vee F) \wedge (T \vee F \vee T) \wedge (F \vee T \vee T) \wedge (F \vee T \vee T) \wedge (F \vee T \vee F) \wedge (T \vee T \vee F)$$

Every clause in 3SAT formula has at least one T literal given the solution

$$= T \wedge T \wedge T \wedge T \wedge T \wedge T$$

$$\boxed{\therefore F' = T}$$

Now let's plug in the same values in our original SAT equation F

$$F = (T \vee F) \wedge (F \vee T \vee T \vee T \vee T \vee T)$$

Every clause in SAT formula has at least one T literal

$$= T \wedge T$$

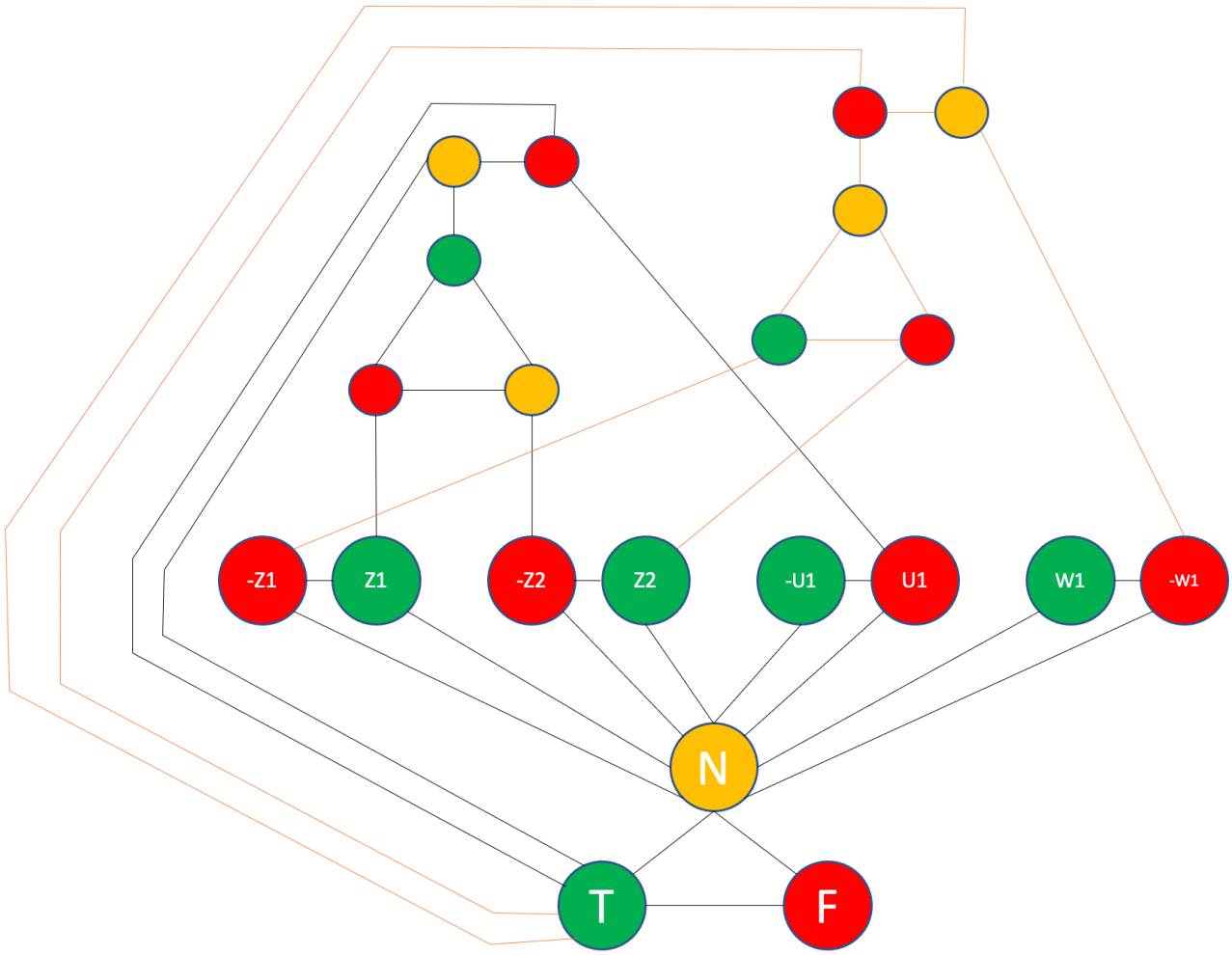
$$\boxed{\therefore F = T}$$

$$\boxed{\Rightarrow F' = F = T}$$

Therefore, the derived 3SAT equation F' is a solution for the given SAT equation F

2) To convert second and third clauses of F' into a 3Col graph.

The respective graph for $(Z_1 \vee \neg Z_2 \vee \neg U_1) \wedge (\neg Z_1 \vee Z_2 \vee W_1)$ is shown below.



Solution for SAT node in graph is determined by plugging in values,

$$Z_1 = \text{T}, Z_2 = \text{T}, U_1 = \text{F}, W_1 = \text{T}$$

Into the SAT formula $(Z_1 \vee \neg Z_2 \vee \neg U_1) \wedge (\neg Z_1 \vee Z_2 \vee W_1)$

$$= (T \vee F \vee T) \wedge (F \vee T \vee T)$$

All clauses have at least one T literal.

Hence this instance of 3COL is a solution to our clauses.

Part 2

The first letter of my surname is in the range S-Z, hence my data files are inst-19.tsp, inst-20.tsp, and inst-7.tsp respectively.

Uniform order-based Crossover:

This will exchange half of the genes from 2 parents to a child in random positions, without gene repetition, trying to main the order of the gene as much as possible from parents.

Let parent A = [A,B,C,D,E,F,G]
Let parent B = [G,A,B,D,C,F,E]
In this method, half random indices will be chosen from parent A, and it will be passed on to the child, at the same positions.
Child = [A,,C,D,,F,]
Now the remaining empty positions will be filled by parent B at the same order possible without repetition of elements.
Child = [A,G,C,D,B,F,E]

Order-1 Crossover :

Let parent A = [A,B,C,D,E,F,G]
Let parent B = [G,A,B,D,C,F,E]
In this method, a random chunk(genes) from A would be removed for the child.
Let's assume that chunk = [C,D,E]
Let child = B = [G,A,B,D,C,F,E]
All the elements from the chunk would be removed from B and called as child
Child = [G,A,B,F]

Now, the chunk can be appended as it is or it can be shuffled.
We are shuffling the chunk before adding it to the end of child.
Shuffled chunk:
Chunk = [D,E,C]
Adding it to the end of child,
Child = [G,A,B,F,D,E,C]

That's how the algorithm works.
A random section will be sliced from the gene from parent A.
At the moment, the child will be a clone of parent B.
All the elements in the sliced section will be removed from the child.
The sliced section will be shuffled and added to the end of the child to create a new crossed over gene.

Inversion Mutation :

In this algorithm, a random section will be sliced from the gene from individual. The sliced section will be reversed and inserted into the individual at the position of the section.
Let Individual A = [A,B,C,D,E,F,G]
In this method, a random chunk(genes) from A would be removed for the child.
Let's assume that chunk = [C,D,E]
Now, the chunk will be reversed,
Chunk = [E,D,C]
It will be added in the same position of the A where it was initially sliced
A = [A,B,E,D,C,F,G]
is the mutated individual.

Scramble Mutation:

In this algorithm, a random section will be sliced from the gene from individual. The sliced section will be shuffled/scrambled and inserted into the individual at the position of the section.
Let Individual A = [A,B,C,D,E,F,G]
In this method, a random chunk(genes) from A would be removed for the child.
Let's assume that chunk = [C,D,E]
Now, the chunk will be scrambled/shuffled,

Chunk = [E,C,D]
It will be added in the same position of the A where it was initially sliced
A = [A,B,E,C,D,F,G]
is the mutated individual.

Binary Tournament Selection:

2 individuals are chosen from the mating pool randomly.
They are both are compared in fitness.
The fitter one will be chosen as a parent.
This process happens again to choose 2 parents and are returned.
A same individual can't be chosen twice, as 2 parents.

Initial Population Nearest Neighbour Selection:

A random point will be selected. From there, it will be the routed through the closest unrouted cities until the last city. This is the initial population.

All the above mentioned crossover and mutation operators have been implemented in python with very high code efficiency and well tested.

The following are the Evaluation of the Genetic algorithms for the given Configurations:

(All the durations are in seconds)

Configuration 1:
Initial Solution: Random
Crossover: Order 1 crossover
Mutation: Inversion Mutation
Selection: Binary Tournament Selection

File Name	Mean duration	Median duration	Mean fitness	Median fitness
inst-19.tsp	97.62	97.38786959648132	17795130.411675684	17795051.147754878
inst-20.tsp	477.66	453.3126883506775	124248208.41416766	123540234.84326749
inst-7.tsp	1144.73	929.7494015693665	308079918.9179949	308110128.7586848

Configuration 2:

Initial Solution: Random

Crossover: Uniform Crossover

Mutation: Scramble Mutation

Selection: Binary Tournament Selection

File Name	Mean duration	Median duration	Mean fitness	Median fitness
inst-19.tsp	98.0839638710022	99.49113988876343	18026724.190690953	17920974.083364345
inst-20.tsp	477.80553278923037	451.8600928783417	123272205.27594177	123209856.86049587
inst-7.tsp	1143.2637591362	927.6741096973419	307309404.21054935	308785070.50812095

Configuration 3:

Initial Solution: Random

Crossover: Order 1 Crossover

Mutation: Scramble Mutation

Selection: Binary Tournament Selection

File Name	Mean duration	Median duration	Mean fitness	Median fitness
inst-19.tsp	98.0839638710022	99.49113988876343	18026724.190690953	17920974.083364345

inst-20.tsp	477.80553278923037	451.8600928783417	123272205.27594177	123209856.86049587
inst-7.tsp	1143.2637591362	927.6741096973419	307309404.21054935	308785070.50812095

Configuration 4:

Initial Solution: Random

Crossover: Uniform Crossover

Mutation: Inversion Mutation

Selection: Binary Tournament Selection

File Name	**Mean duration**	**Median duration**	**Mean fitness**	**Median fitness**
inst-19.tsp	97.84572324752807	95.81364274024963	17950409.464711282	17973783.64275665
inst-20.tsp	478.31813502311707	452.8252799510956	124407201.23883852	124673609.78801471
inst-7.tsp	1144.1912472248077	929.0073041915894	306659844.2483099	305240434.28847957

Configuration 5:

Initial Solution: Heuristic

Crossover: Order 1 Crossover

Mutation: Scramble Mutation

Selection: Binary Tournament Selection

File Name	**Mean duration**	**Median duration**	**Mean fitness**	**Median fitness**
inst-19.tsp	166.9017647266388	166.6010377407074	17792649.866059925	17709729.81318765
inst-20.tsp	786.5336867809295	735.2284455299377	123989426.98685634	123813999.16889316
inst-7.tsp	2048.9242165565493	1558.7671387195587	306413025.98627734	307569408.3246693

Configuration 6:
Initial Solution: Heuristic
Crossover: Uniform Crossover
Mutation: Inversion Mutation
Selection: Binary Tournament Selection

File Name	Mean duration	Median duration	Mean fitness	Median fitness
inst-19.tsp	99.08217053413391	101.04914951324463	17812710.495386768	17910012.640963487
inst-20.tsp	480.17518439292905	457.79233145713806	124254924.07911062	124968027.41415663
inst-7.tsp	1144.5700120925903	927.1536228656769	309075028.2506258	309349623.54359925

Observations and Conclusions

The highest fitness value was observed in configuration 2, with good durations.
Configuration 5 runs the slowest with almost double the time.
Run times of configurations 1 to 4 and 6 are very similar with close fitness values.
They perform very similarly for the specified parameters.

More permutations of combinations of populations and mutation rate have been experimented with the above 6 configurations and have been displayed in TSP_Benchmark_Report.xlsx attached in this folder

Please visit TSP_Benchmark_Report.xlsx since the results couldn't be accommodated in this PDF due to its high volume of data.
Use of Filter option would be highly useful to study specific combinations.

Permutations and combinations of the following have been tested in the above file.
Population sizes: [50,100,150]
Mutation rates: [0.025,0.05,0.1]
Configurations:[1,2,3,4,5,6]

Therefore there are a total of 3*3*9 = 54 results stored in separate folders inside the “Results” folder.

The observation from the spreadsheet is that, the higher the population size, there is slightly higher fitness in some cases.
The 0.1 mutation rate works well giving higher fitness in lesser number of iterations (Observed from the print feedback in terminal)
Lower the mutation, the fitness doesn’t go to its best.
Lower the population, the more number of iterations it is required to achieve nominal fitness levels. (Observed from the print feedback in terminal)

Within the “Results” folder each folder is named as such:

“<Initial population type>_<Selection type>_<Crossover Type>_<Mutation type>_<Population Size>_<Mutation Rate>_<Number of Iterations>”

Within each of those folders are:
Runs of each instance 5 times, benchmark results file for each instance.

Please view each of the python files in the given .zip file which have been well explained.

References:

- CIT MHO Material
- Python documentation

Thanks to Dr.Diarmuid Grimes for teaching me such a wonder concept which I thoroughly enjoyed during the lectures, while building the programs and I’ve done a lot more exploration and programming due to the cultivated interest!