Student Id : R00183682
Student Name: Rishabh Chandaliya

Metaheuristic Optimization.

Solving TSP Problem with Genetic Algorithm

Part 1: NP-completeness

1. If the last digit of your student id is less than 3 use
   F = $(z_1 \vee -z_2) \wedge (-z_1 \vee z_2 \vee z_3 \vee z_4 \vee z_5)$

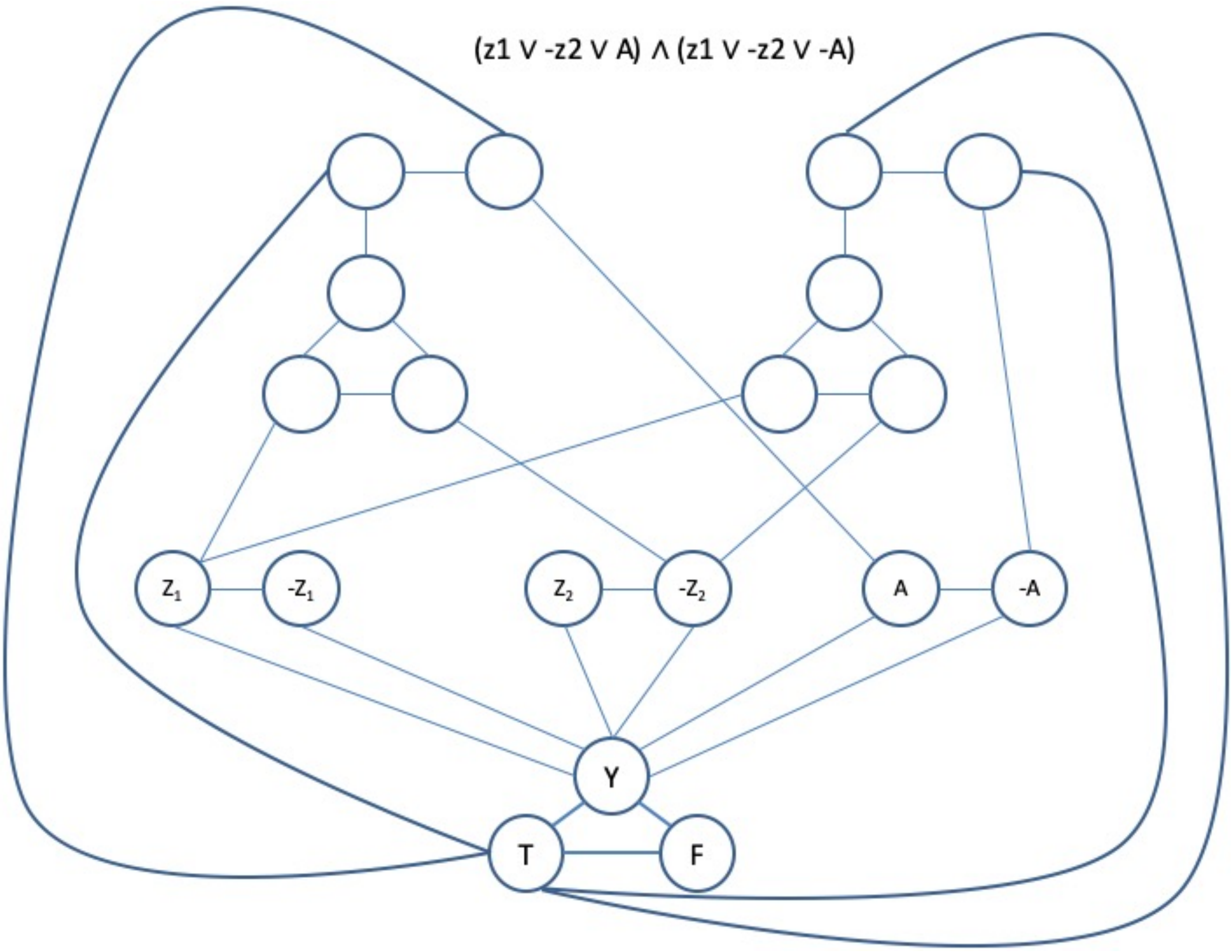
   3SAT formula:

   F' = (z1 V -z2 V A) Λ (z1 V -z2 V -A) Λ (-z1 V z2 V B) Λ (-B V z3 V C) Λ (-C V z4 V z5)

   (z1 V -z2 V A) Λ (z1 V -z2 V -A)

   T F F T F T

2. The first two clauses of F' if the first letter of your first name is in the range J-R



Student Id : R00183682
Student Name: Rishabh Chandaliya

Part 2: Genetic Algorithms

A **genetic algorithm** is a search based on the mechanics of natural selection and genetics. As the name says genetic, it is more related to genes, chromosomes where we get results using the probabilistic mechanism to the question.

We will use genetic algorithm to solve Traveling Salesman Problem (TSP)

To understand TSP, let's take an example and discuss the problem briefly

Imagine you're the manager of logistic company and have to assign delivery work to your courier guy to visit total of 10-12 locations and deliver the courier to the respective house

Before giving him the packages, you'll work out some plan (shortest path) to visit some houses first and then the others based on your previous judgements and experiences. Chances of getting the optimal path are less but not zero. But is it always the best or the optimum path?

For an instance:

- 12 location to visit.
- First selection has 12 location probability, second has 11, third has 10 and so on and so forth in the last selection there is no more location to visit
- So, we can write it has 12! = 12*11*10*9*8*7*6*5*4*3*2*1= 479001600
- Expected number of trials to find optimal solution = 47 million

To solve this problem, it is impractical and also requires modern computing engines. Therefore, we use genetic algorithm to solve difficult optimisation problem.

But we use genetic algorithms when we have a larger search space it would be too naïve to use GA to solve simple solutions.

CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Student Id : R00183682
Student Name: Rishabh Chandaliya

Terminology:

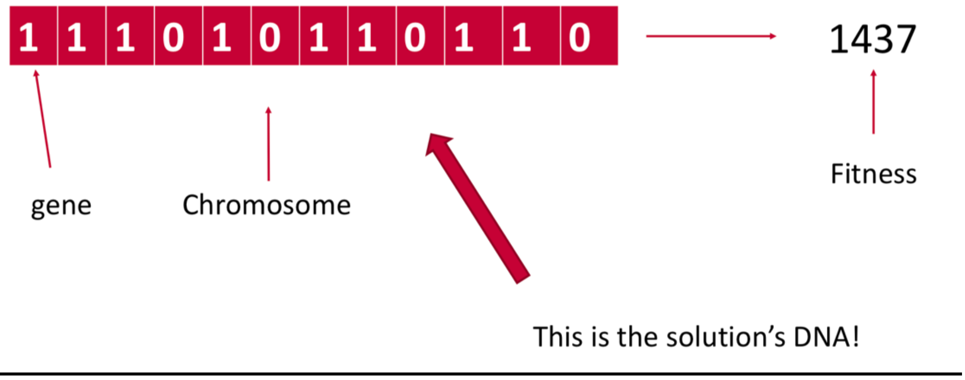


Image ref: Dr. Diarmuid Grimes (Lecture Notes)

1. Population – Collection of chromosomes

2. Chromosome – Collection of genes

3. Fitness value – Cost of chromosome

4. Genes – Cell or variable in chromosome

Basic Structure of GA (Genetic Algorithm):

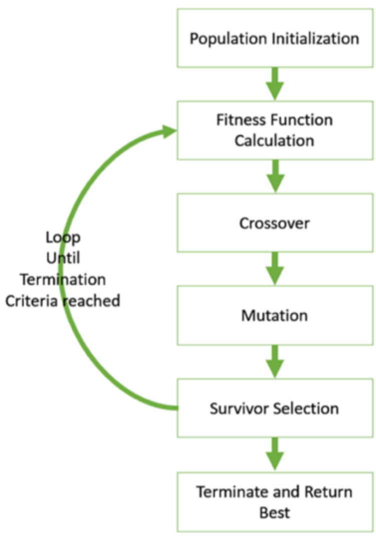


Image ref: Dr. Diarmuid Grimes (Lecture Notes)

1. **Initialization**: Upon initialisation, each individual or chromosome is generated randomly or using heuristic approach and then every individual's fitness value is calculated using the distance between the all the cities which can be calculated using sum of all distances within the permutation

↓ Lesser the Distance (Between Cities) = ↑Higher the fitness value(of individual's)

```
class Individual:
    def __init__(self, num):
        self.fitness = -1
        self.genes = []
        self.genSize = num
        for i in range(0, num):
            self.genes.append( chr(random.randint(32, 128)) )

    def getPhrase(self):
        return "".join(str(x) for x in self.genes)

    def getFitness(self):
        return self.fitness

    def computeFitness(self, target):
        score = 0.0
        for i in range(0, len(self.genes)):
```

Chromosome, e.g., [HeLlo World]

Random values

1

percentage of correct characters

Image ref: Dr. Diarmuid Grimes (Lecture Notes)

<u>Random values</u>: In the above image random numbers are generated from 0 to gensize(where gensize is equal to number of cities)

<u>computeFitness</u>: This function find out the actual cost (Euclidean distance)between two cities. To minimise the cost

2. **Selection:** The overall idea of selection is to make fitter individuals for our new generation neglecting the worst one (not completely) which helps us improving our overall populations fitness. Selection can be done randomly or using some methods (in our case we are using stochastic universal sampling)

```
def GAStep(self):
    matingPool = []
    for gene_i in self.population:
            elementsInPool = int(gene_i.getFitness() * 100)
            for i in range(0, elementsInPool):
                    matingPool.append(gene_i)

    for gene_i in range(0, len(self.population)):
         indexPartnerA = random.randint(0, len(matingPool)-1)
         indexPartnerB = random.randint(0, len(matingPool)-1)
         partnerA = matingPool[indexPartnerA]
         partnerB = matingPool[indexPartnerB]

         child = self.crossover(partnerA, partnerB)
         self.mutate(child)
         child.computeFitness(self.target)

         self.population[gene_i] = child
         if child.getFitness() > self.best.getFitness():
            self.best = child
```

Mating Pool Candidate Individuals

Selection

Update fitness

Image ref: Dr. Diarmuid Grimes (Lecture Notes)

3. **Crossover:** Changing or combining certain genes of 2 individuals to form even fitter children. It's more alike sex how it works in nature between the living beings. With this we get more diverse individuals on which further mutation is done.
4. **Mutation:** To avoid having similar individuals into our population we add very small randomness in which just the single genes of the individual are swapped.

CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Student Id : R00183682
Student Name: Rishabh Chandaliya

```python
class GA:

    #Crossover
    def crossover(self, ind1, ind2):
        child = Individual(self.genSize)
        midPoint = random.randint(0, self.genSize)
        for i in range(0, self.genSize):
            if(i > midPoint):
                child.genes[i] = ind1.genes[i]
            else:
                child.genes[i] = ind2.genes[i]
            return child

    #Mutation
    def mutate(self, ind):
        for i in range(0, self.genSize):
            if(random.random() < self.mutationRate):
                ind.genes[i] = chr(random.randint(32, 128))
```

Midpoint crossover
Half from one and half from the other

Mutation rate

Random modifications for a given gene

Basic Evaluation: Population size = 100 Mutation rate = 0.1

Configuration1

File Name	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5	Mean	Median
inst-0	21360954.2	22145852.61	21902142.08	21893417.09	22007024.24	21861878	21902142.1
inst-5	432363513.4	429708677.7	427132732.6	429902473.3	423650005.5	428551480	427132733
inst-13	110583394.9	111048072.2	109139047	109749183.1	109442875.6	109992515	109139047

Configuration2

File Name	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5	Mean	Median
inst-0	22353895.09	22438255.61	21843227.67	22184690.2	22031247.91	22170263.3	21843227.7
inst-5	433601508.4	432450256.1	429311924.4	425898609	434271165.1	431106693	429311924
inst-13	109610929.9	107071681.4	109790223.2	109991060.4	110766714.3	109446122	109790223

Extensive Evaluation:

Configuration3

File Name	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5	Mean	Median
inst-0	21448303.41	21498488.25	21373261.01	21632847.93	21699011.67	21530382.5	21373261
inst-5	426012034.5	426730611.4	418894173.6	425484091.4	424848617.5	424393906	418894174
inst-13	106577235.4	106699713.1	108168347.7	108808649.4	105762012.7	107203192	108168348

Configuration4

File Name	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5	Mean	Median
inst-0	21339923.29	21765781.77	21546973.93	21528685.04	21807363.12	21597745.4	21546973.9

inst-5	423125141.8	426922219.3	423744425.3	422043768.3	421694397.2	423505990	423744425
inst-13	107462637	107817520.4	106221527.6	104038316.8	107607779.7	106629556	106221528

Configuration5

File Name	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5	Mean	Median
inst-0	22011348.82	21247039.41	21458781.63	21566189.23	21474431.82	21551558.2	21458781.6
inst-5	422916777	426428230	422159252.4	427750041.7	427931549.3	425437170	422159252
inst-13	107479094.2	103920204.1	108794817.1	108778590.3	106907716.7	107176084	108794817

Configuration6

File Name	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5	Mean	Median
inst-0	21482366.82	21467262.59	21866032.63	21612976.43	21436302.65	21572988.2	21866032.6
inst-5	423843467.1	420624584.8	423719283.7	427776422.5	427285889.7	424649930	423719284
inst-13	105431139.4	106373101.7	109511891.3	107547634.8	104049984.8	106582750	109511891

Configuration7

File Name	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5	Mean	Median
inst-0	4126111.864	3992001.98	3992001.98	3992001.98	4126111.864	4045645.93	3992001.98
inst-5							
inst-13	7203070.735	7259774.525	7203070.735	7203070.735	7203070.735	7214411.49	7203070.74

Configuration8

File Name	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5	Mean	Median
inst-0	3992001.98	3992001.98	3992001.98	3992001.98	3992001.98	3992001.98	3992001.98
inst-5							
inst-13	7203070.735	7213226.947	7203070.735	7203070.735	7203070.735	7205101.98	7203070.74

Due to time constrain inst5 file for configuration 7 & 8 did not processed completely. Hence, output could not be showed

References

- Dr. Diarmuid Grimes (Lecture Notes)
- Wikipedia

Student Id : R00183682
Student Name: Rishabh Chandaliya