ANALYSIS OF CROSSOVER, MUTATION METHODS AND RATES OF GENETIC ALGORITHMS APPLIED ON TRAVELING SALESMAN PROBLEM

(GENETİK ALGORİTMALARIN ÇAPRAZLAMA, MUTASYON METODLARININ VE PARAMETRELERİNİN GEZGİN SATICI PROBLEMİ ÜZERİNDE ANALİZİ)

by

Adnan BAL, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

Nov 2018

This is to certify that the thesis entitled

ANALYSIS OF CROSSOVER, MUTATION METHODS AND RATES OF GENETIC ALGORITHMS APPLIED ON TRAVELING SALESMAN PROBLEM

prepared by **Adnan BAL** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering** at the **Galatasaray University** is approved by the

Examining Committee:

Asst. Prof. Dr. Murat AKIN (Supervisor)
Department of Computer Engineering ----------
Galatasaray University

Asst. Prof. Dr. Burak PARLAK
Department of Computer Engineering ----------
Galatasaray University

Asst. Prof. Dr. İlker ÜSTOĞLU
Electrical & Electronics Faculty Control and ----------
Automation Engineering Department
Yıldız Technical University

Date: ----------

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Asst. Prof. Murat Akın, for his helpful advices and great patience during the process.

I also would like to thank my family, for supporting me to create this special work of mine.

Nov, 2018
Adnan BAL

TABLE OF CONTENTS

LIST OF SYMBOLS

EP	**:** Evolutionary Programming
GA	**:** Genetic Algorithm
GP	**:** Genetic Programming
OBX	**:** Order Based Crossover
PBX	**:** Position Based Crossover
PMX	**:** Partially Mapped Crossover
TSP	**:** Traveling Salesman Problem

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

With the rapid development of whole industry(automotive and especially logistics) and software industry, increasing demand by customers and supply by manifacturers led the optimization more and more important nowadays. By the word for optimization, we mean minimizing production times, maximizing product logictics per transportation or minimizing fuel usage/maximizing fuel saving/efficiency for transportation vehicles. By the demand of these optimizations by the industry, also led optimization algorithms/techniques to grow and evolve. With the evolution of computers and computation powers, classic optimization techniques also evolved. One of evolutinary optimization techniques, Genetic algorithms and genetic programming, corresponded to these heavy demand of optimization area. Basically, genetic algorithms evolved from genetics and applications of sir Charles Darwin, crossover and mutation principles.

Using Genetic Algorithms, we have the ability to optimize our solutions for hard problems. Simply, finding/choosing random solutions to the problem and make crossover and mutations on these solutions as the nature does. Crossing over and mutate the parts of solutions by switching the meaningful data between solutions and hope to reach to the best optimized solution. Generally we reach to the optimized solution by finding and trying correct or better crossover and mutation rates. In other words, choosing bad rates for these parameters, most likely leads to worse optimization.

In this work, firstly, we presented the genetic algorithms in general way and after that we go in deep and used genetic algorithms to find better optimized results for the famous Traveling Salesman Problem. We chose to apply genetic algorithms on geographical regions of Turkey(Marmara, Aegean and Black Sea regions, 32 cities in total) to find best or best optimized route to travel. While applying genetic algorithms,

we modified crossover methods, mutation methods and crossover and mutation rates to reach to the best possible route and analysed final solutions for each used parameter/method and made a comparison between them.

Finally, we presented the compared results on graphics to visualize the evolution for each presented parameter. By making these research, we aim to reach out the best or better parameters for real use cases used in the logistics industry to reach better fuel efficiency and reducing fuel costs.

RÉSUMÉ

Avec le développement rapide de toute l'industrie (automobile et surtout logistique) et de l'industrie du logiciel, l'augmentation de la demande des clients et l'approvisionnement par les fabricants ont conduit l'optimisation de plus en plus importante de nos jours. Par optimisation, on veut dire qu la minimisation des temps de production, la maximisation de la logistique du produit par transport ou la minimisation de la consommation de carburant / la maximisation de l'économie de carburant / efficacité pour les véhicules de transport. Par la demande de ces optimisations par l'industrie, a également conduit des algorithmes / techniques d'optimisation pour grandir et évoluer. Avec l'évolution des ordinateurs et des puissances de ses calcul, les techniques classiques d'optimisation ont également évolué. L'une des techniques d'optimisation évolutive, les algorithmes génétiques et la programmation génétique, correspondaient à cette forte demande de domaine d'optimisation. Fondamentalement, les algorithmes génétiques ont évolué à partir de la génétique et des applications de sir Charles Darwin, les principes de croisement(crossover) et de mutation.

En utilisant des algorithmes génétiques, nous avons la capacité d'optimiser nos solutions pour les problèmes difficiles. Simplement, trouver / choisir des solutions aléatoires au problème et faire des croisements et des mutations sur ces solutions comme le fait la nature. Traverser et muter les parties des solutions en changeant les données significatives entre les solutions et espérer trouver la solution la mieux optimisée. Généralement, on trouve la solution optimisée en trouvant et en essayant des taux de croisement et de mutation corrects ou meilleurs. En d'autres termes, le choix de mauvais taux pour ces paramètres conduit très probablement à une optimisation moins bonne.

Dans ce travail, nous avons d'abord présenté les algorithmes génétiques de façon générale et ensuite nous allons dans des algorithmes génétiques profonds et utilisés pour

trouver des résultats mieux optimisés pour le fameux Traveling Salesman Problem. Nous avons choisi d'appliquer des algorithmes génétiques sur les régions géographiques de la Turquie (régions de Marmara, de la mer Egée et de la mer Noire, 32 villes au total) pour trouver le meilleur ou le meilleur optimisé pour voyager. En appliquant des algorithmes génétiques, nous avons modifié les méthodes de croisement, les méthodes de mutation et les taux de croisement et de mutation pour atteindre la meilleure route possible et analysé les solutions finales pour chaque paramètre / méthode utilisé et fait une comparaison entre eux.

Enfin, nous avons présenté les résultats comparé sur les graphiques pour visualiser l'évolution de chaque paramètre présenté. En faisant ces recherches, nous cherchons à atteindre les meilleurs ou les meilleurs paramètres pour les cas d'utilisation réels utilisés dans l'industrie de la logistique pour atteindre une meilleure efficacité énergétique et réduire les coûts de carburant.

ÖZET

Tüm endüstrinin(otomotiv ve özellikle lojistik alanında) ve yazılım endüstrisinde ki hızlı gelişmelerle, artan müşteri talepleri ve üreticilerin arzları sayesinde optimizasyon her gün daha çok önem kazanmaktadır. Optimizasyonla kastımız, üretim zamanlarının düşürülmesi, ürün lojistiğinin artırılması veya taşıma maliyetlerinde yakıt tüketiminin düşürülmesi anlatılmak istenmektedir. Endüstrinin bu tarz optimizasyon talepleri ayrıca optimizasyon algoritmalarının/tekniklerinin gelişerek evrilmesine katkıda bulunmuştur. Bilgisayarların gelişimi ve hesap kabiliyetlerinin gelişmesiyle, klasik optimizasyon teknikleri de evrilmiştir. Evolutionary optimizasyon tekniklerinden olan Genetik algoritmalar ve genetik programlama, optimizasyon alanında ki yüklü talebe yanıt vermeye çalışmaktadır. Temel olarak, generik algoritmalar, biyolojik genetik ve Sir Charles Darwin'in genetik alanında ki çaprazlama ve mutasyon uygulamalarından türetilmiştir.

Genetik algoritmaları kullanarak, zor problemlerin daha optimize edilmiş sonuçlarına daha kolay olarak ulaşma şansına sahip oluruz. Genel olarak, aynı doğada olduğu gibi, rastgele çözümler bularak/seçerek, bu sonuçlara çaprazlama ve mutasyon teknikleri uygulayarak daha optimize sonuçlar bulmayı hedefliyoruz. Rastgele sonuçları birbiri arasında, parçalı olarak anlamlı verilerini çaprazlama ve mutasyon uygulayarak, daha optimize edilmiş sonuçlara varmayı umuyoruz. Genellikle, optimize edilmiş sonuca, doğru ve daha iyi çaprazlama ve mutasyon oranları seçerek ulaşmayı deniyoruz. Diğer bir deyişle, parametreler için kötü oranlar seçmek, bizi çoğunlukla daha kötü ve ya optimize olmayan sonuçlara ulaştıracaktır.

Bu çalışmamızda, öncelikle, genetik algoritmaların genel konseptlerini tanıtıyoruz, daha sonra ise daha derine inerek ve spesifik şekilde genetik algoritmaları kullanarak, ünlü Gezgin Satıcı Problemi'ne optimize çözümler arıyoruz. Bu çalışmamızda, genetik

algoritmayı Türkiyenin coğrafi bölgelerine(Marmara, Ege ve karadeniz bölgelerindeki şehirler, toplam 32 şehir) uygulayarak bu şehirler arasında ki en kısa yolu bulmaya çalışıyoruz. Genetik algoritmayı uygularken, çaprazlama metodlarını, mutasyon metodlarını, çaprazlama ve mutasyon oranlarını değiştirerek, en optimize yolu bulmaya çalışıyoruz ve sonuç olarak bulunan optimize sonuçları tüm bu parametler için ayrı ayrı analiz edip ortaya koyuyoruz.

Sonuçta ise, karşılaştırılmış sonuçları grafik üzerinde göstererek, her parametrenin sonuca ne denli etki ettiğini ortaya koyuyoruz. Bu araştırmayı yaparak, gerçek hayatta lojistik endüstrisinde de aktif olarak kullanılan use caseler için daha doğru parametrelerin seçimine katkıda bulunarak, şirketlerin daha iyi yakıt tasarrufu elde etmelerine katkı sağlamayı amaçlıyoruz.

1. INTRODUCTION

Nowadays, in the industry, heavy demand of customers for products is known. Everyday, the consumption of products rises to significant levels. This consumption rise pushes manufacturers to calculate their every expense and to make improvements on every step of production and logictics. Which is why, optimization takes place and help manufacturers to improve their work and profitability. This need of optimization leads manufacturers to search for ways to improve their work and also that leads to find better algorithms that suits for their needs. Being one of evolutionary algorithms, Genetic Algorithms and Genetic Programming, take into action at this point.

With the evolution of computers and their computational power, every calculation and optimization techniques are also evolving. This evolution also leads algorithms to improve. Evolution-based systems and evolution programs(EP) has been around since 1960s and evolutionary algorithms are also improving every day. The word evolutionary is coming from evolution in biology, from works of Sir Charles Darwin, as species evolve with cross-overs and mutations to their childs/generations in time naturally. This is done by natural selection by the nature. Cross-over is simply transmitting parent's meaningful data to their childs, and mutation is changing or deterioration of one or various meaningful data of a child. These kind of operators provide variation of offsprings and also for the population. By this way, better offsprings and thus better populations are created.

Using this evolution concept, we have the ability to apply this concept to computer science by creating parents and evolve them and create better offsprings and result. This concept allows us to reach better and better results for optimization. Genetic algorithms are heavily used for optimization in the industry and research area to achieve better optimization results.

Especially, for the logistics industry, the production times and transportation costs play significant role for their profitability. The manufacturers always try to achieve better results, to find better and optimized routes for logistics and for their profits. They try to achieve, primarily maximazing production, minimizing cost and finally maximize the profits. Achieving this, they have to evaluate their past productions or calculate their best shortest route for logistics. Optimization is stated in Chong et al[2008] as ;

"*Optimization is central to any problem involving decision making, whether in engineering or in economics. The task of decision making entails choosing between various alternatives. This choice is governed by our desire to make the "best" decision."*

Genetic Algorithms are one of evolutionary algorithms that uses some heuristics and using crossover and mutation methods to achieve better optimization results, when the calculation is heavy. Simply, choosing several random results and crossing over and mutating these results between them and collecting better results to reach to the optimized solution according to their fitness to the problem. In Parmal et al.[2017], genetic algorithms are described as ;

"Genetic Algorithm(GA) is a heuristic search technique for optimization, where it is not possible to analytically establish the extreme of the function. It employs a strategy based on the theory of natural selection to obtain iterative refinement of a population of potential solutions. It has been applied to diverse fields in problems like Traveling salesman problem (TSP), Marketing, finance etc."

In this work, we analyse Traveling Salesman Problem for cities of 3 different regions of Turkey(Marmara, Aegean and Black Sea regions) that traveling salesman starts at one city and travels each cities and returns back to the starting city. Our aim is to analyse how the genetic algoritm and their parameters affect optimal solutions while trying to find the optimal route. We present each parameter in detail and present each of their behaviour for the results. We modify crossover, mutation methods and crossover, mutation rates. Then we visualize these distinct result and present on the graphics.

Finally, we compare these results to identify which parameters are better choice for better optimization.

2. LITERATURE REVIEW

Genetic algorithms have variety of use cases, and wide area of use. Especially a lot of work has been done for the financial forecasting area and optimization areas in the real life applications. Allen et al.[1999] used genetic algorithms to learn technical trading rules for the S&P 500 using daily price movements from 1928 to 1995. They try to find trading rules for buy/sell orders on the index but it is indicated that after the transaction costs, the found trading rules do not earn excessive returns. Also Parmal et al. [2017] dealt with the problem of optimization of sales of a company, with using available data of a company, applying genetic algorithm to customer and product categories to to find optimal combinations. Kim et al[2005] used genetic algorithms to neural networks for customer targeting as identifying and profiling households. Lin et al.[2004] used sub-set values for parameters instead of single value to generate better optimization for financial buy/sell signals, using sub-set method, better optimization values obtained in result. In financial markets, calculating and analysing historical price data is crucial, so generally researches are highly focused on analysing the market data and to have a concrete meaningful strategies/results.

While the genetic algorithms is an interesting an practical area of research, also general concept researches and practical use case researches have been made widely. In book « Practical Genetic Algorithms , Haupt[2004]» general concepts of optimization and genetic algorithms have been presented as genetic algorithm methods and practical use cases. Bethke[1980] presented the genetic algorithms as function optimizers in his research, this research is also interesting as this research is one of the primary researches in this area. Michalewicz[1992], in his book he presented genetic algorithms in detail, briefly explains GAs and classic problems that can be solved/optimized using GAs as prisoners dilemma and traveling salesman problem.

There are also special and famous optimization problems as Prisoners Dilemma and Traveling Salesman Problem(TSP), heavy number of researches also have been made through the area. The researchers tried to optimize these problems to have better results. Grefenstte et al.[1985] discussed representation methods as ordinal representation and adjacency representations as well as the crossover and also presenting subtour chunking operator , an off-spring is constructed from to parent tours as follows : "First choose a subtour of random length from one parent, then extend the partial tour by choosing a subtour of random length from the other parent. Continue extending the tour by choosing subtours from alternating parents. During the selection of a subtour from a parent, if the parent's edge would introduce a cycle into a partial tour, then extend the partial tour by a random edge which does not introduce a cycle. Continue until a complete tour is constructed." Also Potvin[1996] also discussed several advanced crossover , mutation techniques as partially-mapped crossover(PMX), order based crossover(OBX), position based crossover (PBX) and presented computational results and made comparisons between methods. He also calculated CPU calculation times.

There are also researches for parameters of the genetic algorithms to reach better optimization results by choosing better control parameters. Mills et al.[2014] defined an experiment design and analysis method to determine the relative importance and most effective setting for each control parameter in a GA. Also Boyabatlı et al.[2004] analysed the effect of numerical parameters on the performance of GA based mimulation optimization applications with experimental design techniques. Rexhepi et al.[2013] presented an analysis on impact of parameter values for Traveling Salesman Problem(TSP).

3. GENETIC ALGORITHMS

Genetic Algorithms are one of evolutionary algorithms that use some heuristics and using crossover and mutation methods to achieve better optimization results, when the calculation is heavy. Evolution starts with randomly selected population members, for the mathematical functions, it is randomly generated solutions or results. Simply, choosing several random solutions/results and crossing over and mutating these results between them and collecting better results to reach to the optimized solution according to their fitness to the problem. Figure 3.1 shows the general flow chart of genetic algorithm(GA). In Figure 3.2, pseudo-code for GA is given. Also, in Parmal et al.(2017), genetic algorithms are described as ;

"Genetic Algorithm(GA) is a heuristic search technique for optimization, where it is not possible to analytically establish the extreme of the function. It employs a strategy based on the theory of natural selection to obtain iterative refinement of a population of potential solutions. It has been applied to diverse fields in problems like Traveling salesman problem (TSP), Marketing, finance etc."

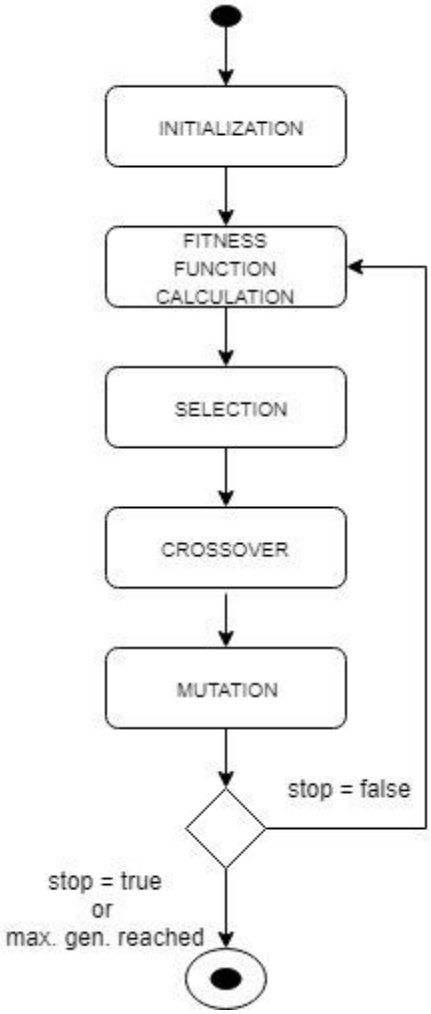

Figure 3.1: Genetic Algorithm Flow Chart

```
function GeneticAlgo()
{
      initialize population
      calculate fitness of population

   while(max. iteration reached or termination condition reached) do
      select parents
      apply crossover with given probability
              apply mutation with given probability
      fitness calculation

      generate new population
   end while
}
```

Figure 3.2: Genetic Algorithm pseudo-code

3.1 OPTIMIZATION

Optimization is selecting a best element or a best solution (regarding some fitness or criterion) in a set of alternatives. Optimization can be done to maximizing or minimizing the solutions according to the problem type. Simply, systematically choosing input values from allowed range and maximize or minimize the solution. Figure 3.2 shows the optimization logic.

Figure 3.3: Optimization Logic

In simple words, we try to find the better/best input values to find better/best output values, better output values(results) mean optimization of the problem. If there is no further better result, that result is called optimal value.

Especially, for the logistics industry, the production times and transportation costs play significant role for their profitability. The manufacturers always try to achieve better results, to find better and optimized routes for logistics and for their profits. They try to achieve, primarily maximazing production, minimizing cost and finally maximize the profits. Achieving this, they have to evaluate their past productions or calculate their best shortest route for logistics.

Mathematically, optimization can be defined as ;

*Given: a function f : **A** → **R** from some set A to the real numbers*

Sought: an element x_0 in A such that $f(x_0) \leq f(x)$ for all x in A ("minimization") or such that $f(x_0) \geq f(x)$ for all x in A ("maximization").

f : objective function to be optimized.

3.2 WHY GENETIC ALGORITHMS ?

Genetic Algorithms provide « good-enough » solutions « fast-enough ». In other words, we may have better optimization fast enough using GAs. This benefit of GAs, make them very attractive especially for hard problems.

3.2.1 SOLVING DIFFICULT PROBLEMS

There are a lot of problems in computer science that requires a lot of computational power, even it takes years to solve these kind of problems. At that point, Genetic Algorithms provide usable near-optimal solutions in a short period of time. That makes GAs more and more attractive.

3.2.2 PROVIDE GOOD SOLUTION FAST

Various difficult problems like Traveling Salesman Problem(TSP), are used in real-life applications like Navigation apps. Thus, providing good-enough solutions fast-enough is very important for such cases. GAs provide required fast and good-enough solutions.

3.3 GENETIC ALGORITHM ADVANTAGES / DISADVANTAGES

3.3.1 ADVANTAGES

Here is a list of advantages of genetic algorithms ;

- Faster and works efficiently than the traditional methods.
- Parallel capability/computing.
- Optimizes both continuous and discrete functions.
- Always provides an answer, solutions get better with better parameter choices.
- Generally useful when the search space and the parameter number is large.

3.3.2 DISADVANTAGES / LIMITATIONS

GA also has some disadvantages/limitations, here are some examples ;

- For simple functions/problems, using GA might be useless or redundant.
- Fitness values for chromosomes are calculated repeatedly, that process might take time and might be expensive in terms of computation.
- There are no guarantees that the found solutions is optimal or its quality.
- If chosen parameters are not good enough or if there is a problem of implementation, then GA might not converge to the optimal solution.

3.4 APPLICATION AREAS

Genetic algorithms have a wide area of application in real life. You may find some application areas of GAs.

- Optimization : Solving optimization problems.
- Economics : GAs also have a part of economics to modelize or characterize price movements for better profits.
- Neural Networks : Training neural networks.
- Paralellization.

- Image Processing : Used for Digital Image Processing, dense pixel matching.
- Vehicle Routing Problems .
- Scheduling Applications : Optimizing for schedule problems, especially time tabling problem.
- Machine Learning : Genetics based machine learning.
- Parametric Design of Aircrafts : Varying the parameters, design of aircrafts gets better results.
- Logistics : Traveling Salesman Problem and its application areas.

3.5 TERMINOLOGY

- **Population :** Subset of some possible solutions (encoded) to the problem. Poulation changes over time with new generations(offsprings).
- **Chromosomes :** One solution to the given problem. An element of the population.
- **Fitness Function :** Specific to the given problem. Fitness function is basically measures how fit / suitable is the solution. It takes solution as the input and generates its suitability as the output.
- **Genetic Operators :**
    - **Crossover :** Exchanging / Transmitting meaningful data of chromosomes between them in a given order.
    - **Mutation :** Changing one or several meaningful data of a chromosome to provide variety in a given probability.
    - **Survivor Selection :** Selection of better solutions to achieve better optimal results.

3.6 REPRESENTATION OF CHROMOSOMES

First and the most important decision for a genetic algorithm is to choose correct representation for our solutions. It is proven that bad representation choice often leads to bad GA performance. In this section, we present various GA representation methods, please keep in mind that representation is problem specific.

3.6.1 BINARY REPRESENTATION

In this type of representation, each data of a chromosome consists of a bit string (0 or 1), it is useful when the solution space is boolean variables – true or false. (e.g. Knapsack problem, 1 represents an item is picked and 0 represents an item is not picked.). Figure 3.4 shows a chromosome consists of binary representation.

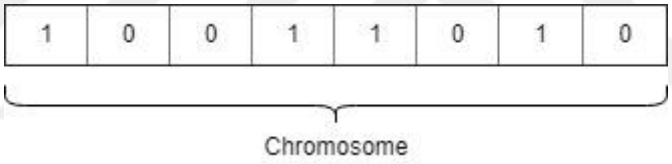

Figure 3.4: Binary Representation Example

3.6.2 INTEGER REPRESENTATION

When we have more solution space than true or false, we may choose to use integer representation, simply represent various terms into integers, e.g. {up, down, left, right} to : {1,2,3,4}. Figure 3.5 shows a chromosome consists of integer representation.

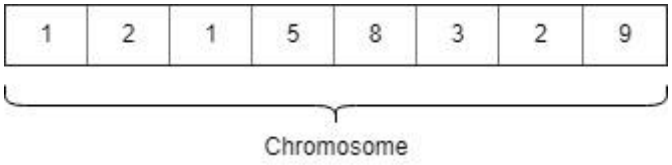

Figure 3.5: Integer Representation Example

3.6.3 PERMUTATION REPRESENTATION

In this representation, solution is represented by order of elements in the list, e.g. Traveling Salesman Problem(TSP), representing order of all the cities which the traveling salesman will visit, makes sense to this problem. Figure 3.5 shows a chromosome consists of permutation representation.

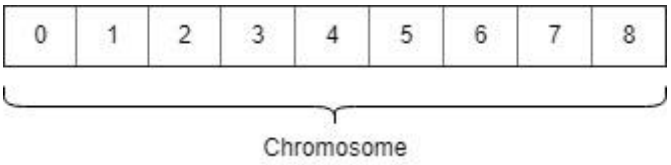

Figure 3.6: Permutation Representation Example

3.7 POPULATION INITIALIZATION

Population is the subset of solutions(chromosomes) in the current generation. Some points are very critical when populating GA optimization ;

- Diversity of the population : if diversity is not good enough, process might lead to premature convergence, thus local maximum or local minimums.
- Population size : this parameter is critical for the performance of the GA, small population size might not be enough for good mating, while very large population size might lead GA to slow down.

3.7.1 INITIALIZATION METHODS

There are several methods to initialize population, two important methods are the following :

- Random Initialization : completely random population of solutions.
- Heuristic Initialization : populate population using a known heuristic for the recent problem.

Using heuristic initialization may cause the population to have similar chromosomes and thus it can deteriorate the diversity and finally it may affect the optimality.

3.8 FITNESS FUNCTION

Fitness function is, simply, a function that takes solutions as input and calculates their fitness values to how *fit* they are to the given problem. It is a measure of « how good a solution » is. Fitness value choice and implementation is very important for a GA, because the fitness function is calculated repeatedly and must be fast enough. In our work, we use **total distance travelled** between cities in a chromosome and then comparing these fitness functions to pick the best chromosomes.

Figure 3.7 shows a real example of our results, there is a population consists of 500 chromosomes and we show 4 last chromosomes and their fitness values, in our case, fitness values is the same as objective function : total distance between cities. Less distance value is better.

```
[ Kayseri Edirne Izmir Yozgat Afyon Bilecik Istanbul Karaman
Yalova Kirikkale Aksaray Usak Sakarya Mugla Kirklareli Manisa
Eskisehir Denizli Tekirdag Balikesir Kirsehir Kocaeli Kutahya Nigde
Sivas Aydin Nevsehir Konya Cankiri Bursa Ankara Canakkale ]
Distance: 14055

[ Mugla Ankara Manisa Kocaeli Nigde Aksaray Eskisehir Izmir Afyon Edirne
Denizli Istanbul Yozgat Nevsehir Yalova Konya Canakkale Sivas Sakarya
Bursa Tekirdag Kirklareli Aydin Kayseri Kutahya Kirsehir Balikesir
Bilecik Usak Karaman Kirikkale Cankiri ]
Distance: 13070


[ Kayseri Bursa Konya Eskisehir Kirikkale Canakkale Yozgat Kirklareli
Istanbul Cankiri Denizli Kutahya Sivas Edirne Nevsehir Ankara Balikesir
Nigde Sakarya Kocaeli Aksaray Usak Yalova Tekirdag Izmir Aydin Kirsehir
Karaman Afyon Bilecik Manisa Mugla ]
Distance: 11913


[ Aksaray Denizli Bilecik Usak Afyon Edirne Cankiri Kayseri Manisa
Karaman Kocaeli Kirklareli Izmir Kirikkale Nigde Nevsehir Bursa Kirsehir
Yozgat Canakkale Aydin Eskisehir Yalova Konya Balikesir Mugla Tekirdag
Istanbul Ankara Sakarya Sivas Kutahya ]
Distance: 12257
```

Figure 3.7: Example Fitness Calculation Results

3.9 PARENT SELECTION

Parent selection process is to select parents that will mate and generate off-springs with data interchange. It is very important to find balance while selecting parents, because always selecting best parents may lead to loss of diversity and that also may lead to premature convergence. So, keeping good diversity is always something to keep in mind for the performance of the GA. (e.g. with loss of diversity means that similar chromosomes will be generated in the next generations, that is not a desirable result for GAs.). Parent selection methods are as follows ;

3.9.1 TOURNAMENT SELECTION

Tournament selection is simply choosing « k » elements from the population at random and compare them and finally, select the best one to become a parent. The same process is repeated for other parents. In our work, we use tournament selection for our selection process.

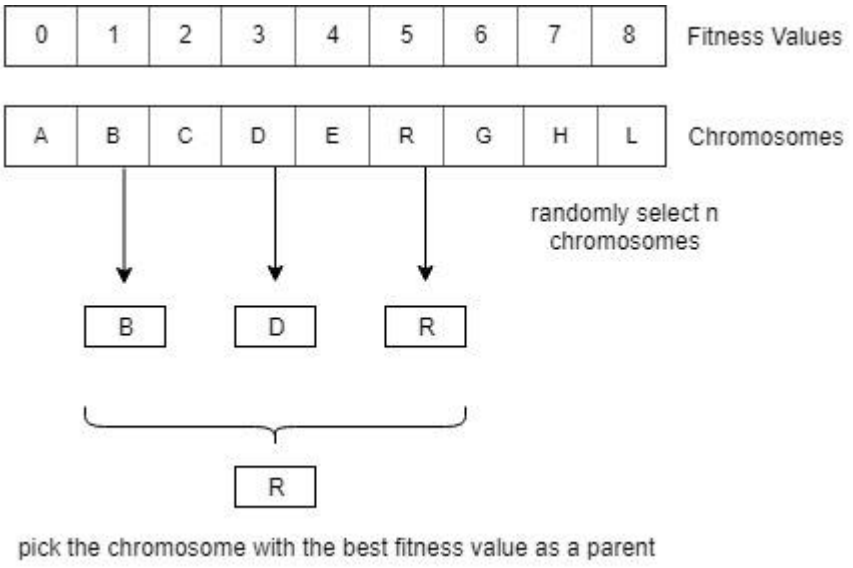

Figure 3.8: Example Fitness Calculation Results

3.9.2 ROULETTE WHEEL SELECTION

In this method, a circular wheel divided into n pies where the n is the number of chromosomes in the population. So, each chromosome has a portion of proportional to its fitness value. By this way, better chromosomes have more chance to be picked as parents.

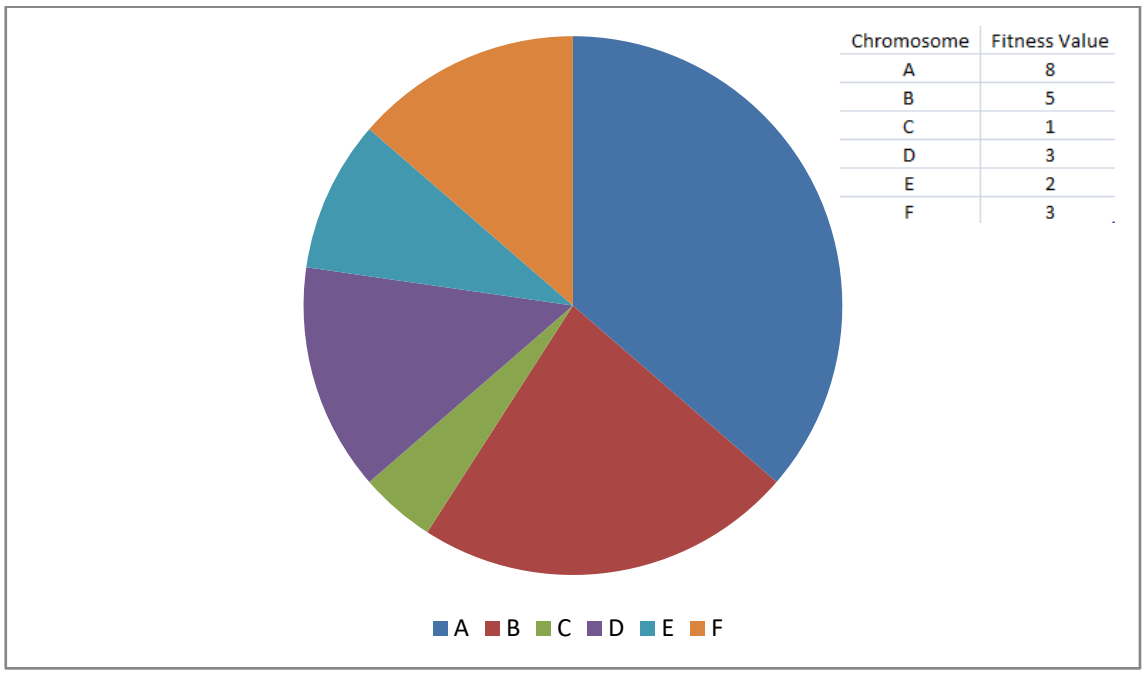

Figure 3.9: Roulette Wheel Selection

3.10 GENETIC OPERATORS

3.10.1 CROSSOVER

Crossover operator is just like biological crossover, interchanging meaningful data between parents, so new generations have properties from both parents. Applying crossover, we have to choose minimum 2 parents and one or more parent can be produced from these parents. Generally applied with a probability of $-\mathbf{P_c}$. For the brevity of the work, we presented 3 crossover methods in detail but there are also a lot

of various crossover methods like Partially Mapped Crossover (PMX), Order Based Crossover(OBX) etc.

3.10.1.1 ONE-POINT CROSSOVER

This crossover operator is done as follows, randomly a crossover point is selected and according to that point, data of parents are swapped to generate new off-springs.

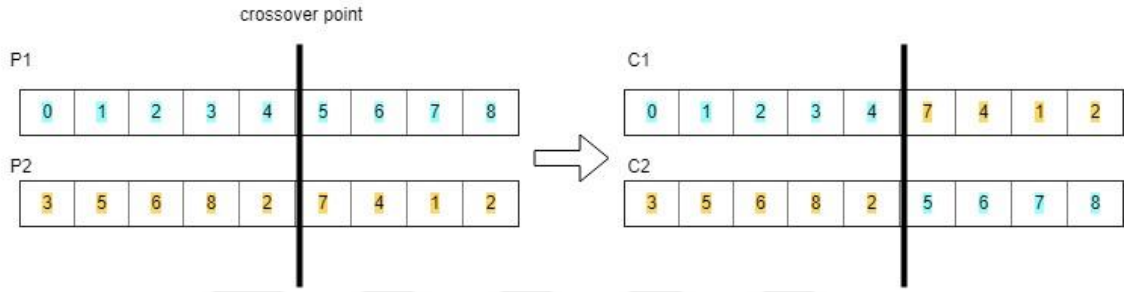

Figure 3.10: One-Point Crossover

3.10.1.2 MULTI POINT CROSSOVER

Randomly choose more than one crossover point and swap alternatingly data from parents to generate new generations.

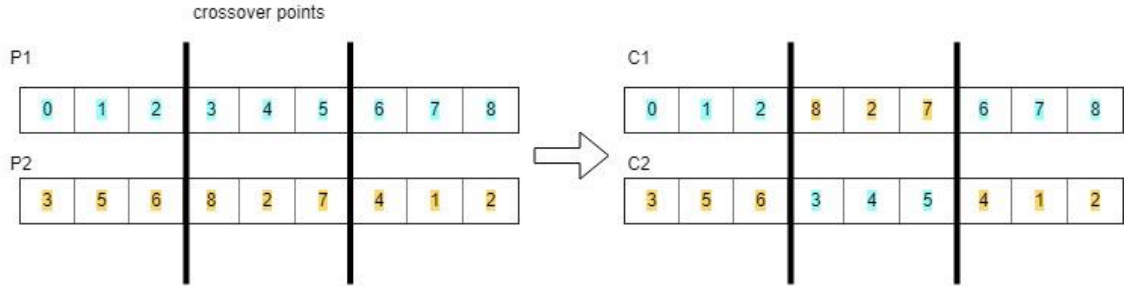

Figure 3.11: Multi Point Crossover

3.10.1.3 UNIFORM CROSSOVER

In this crossover method, we do not use separation segments, instead we treat each gene separately. In simple words, we choose random probability for each gene if this gene will be in the next generation.

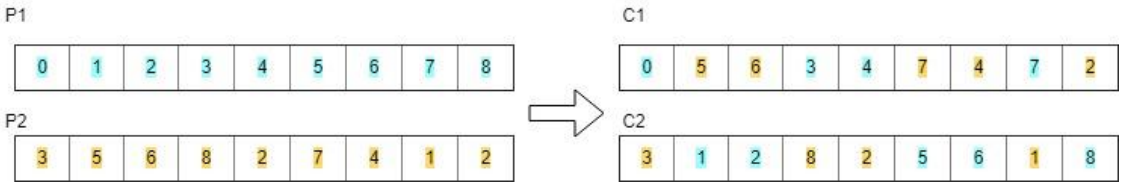

Figure 3.12: Uniform Crossover

3.10.2 MUTATION

Mutation is simply a disorder or defect on a data. Generally applied for maintaining the diversity on the population with a low probability –P_m. If P_m is not low enough, then GA may turn into a random search.

3.10.2.1 INSERTION MUTATION

Choose a random element in the chromosome and insert it into a random place.

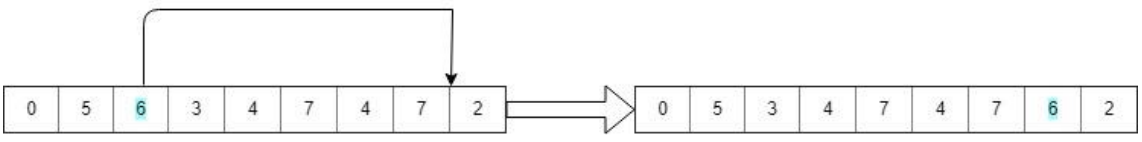

Figure 3.13: Insertion Mutation

3.10.2.2 SWAP MUTATION

Choose random two positions and swap these two genes. Generally used in permutation representations.

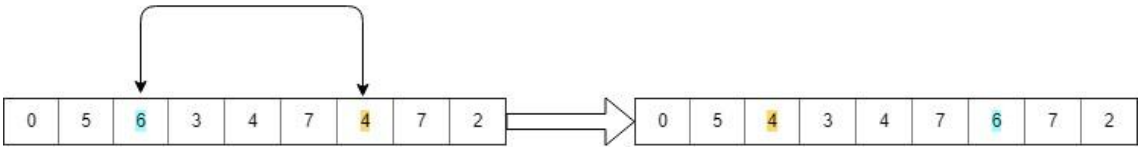

Figure 3.14: Swap Mutation

3.10.2.3 SCRAMBLE MUTATION

In this mutation method, a subset of chromosome is selected and their elements are scrambled at random.

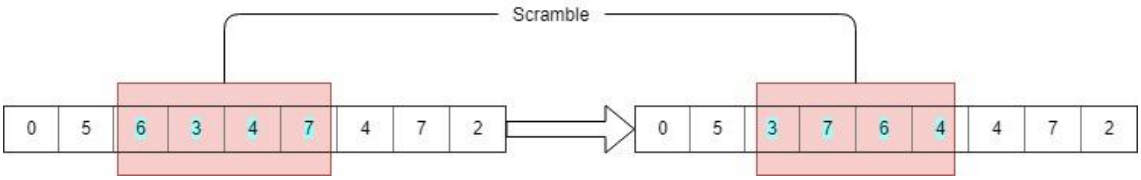

Figure 3.15: Scramble Mutation

3.11 SURVIVAL SELECTION

Survival selection process is to decide which chromosomes will propagate to the next generation and which ones will be removed from the population. It is very important to keep better/fitter chromosomes in the population and also we have to maintain the diversity in the population.

For keeping the fitter chromosomes in the population, generally, the ***Elitism Method*** is used. It simply keeps last best chromosome in the population and directly propagates it

to the next generation. Randomness can be used to decide which ones to remove and propagate, but it affects the performance, thus, following methods are used to maintain better performance ;

3.11.1 FITNESS BASED SELECTION

In this method, fitnesses of the chromosomes play critical role. Basically, least fit chromosomes are replaced by the new off-springs. Sometimes, it is better to use some randomness to maintain diversity. Figure 3.14 shows the fitness based selection method.

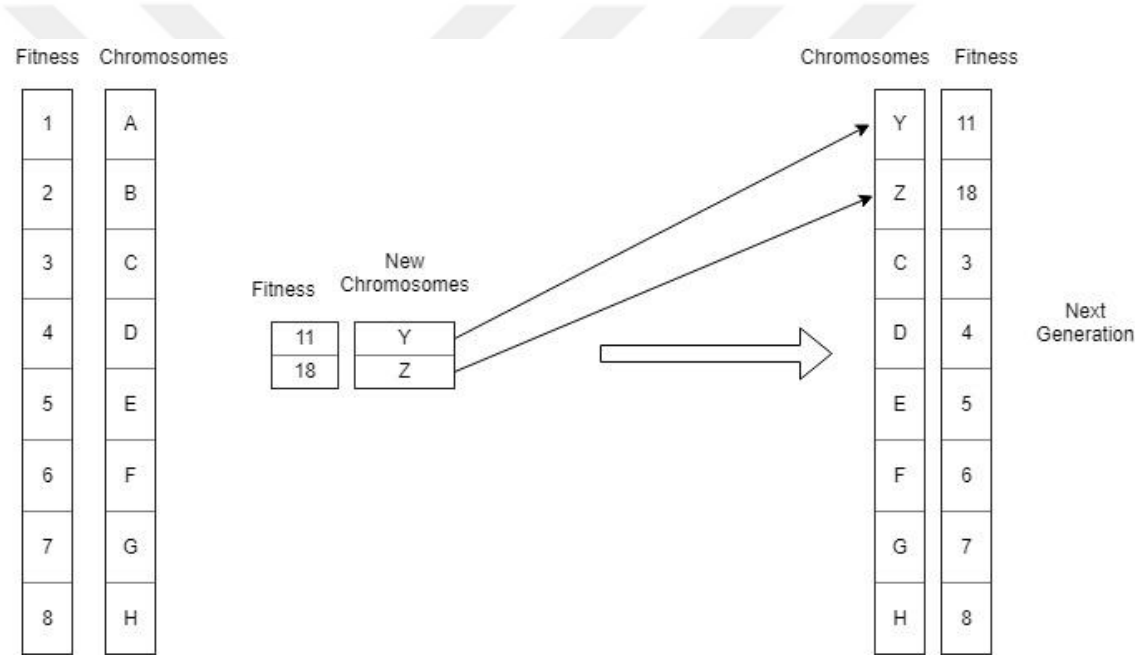

Figure 3.16: Fitness Based Selection

3.11.2 AGE BASED SELECTION

In age based selection, fitness of the chromosomes does not play a role, instead we use age of the chromosomes that for how many generations a chromosome has been in this population. The older chromosomes are replaced by the new off-springs.

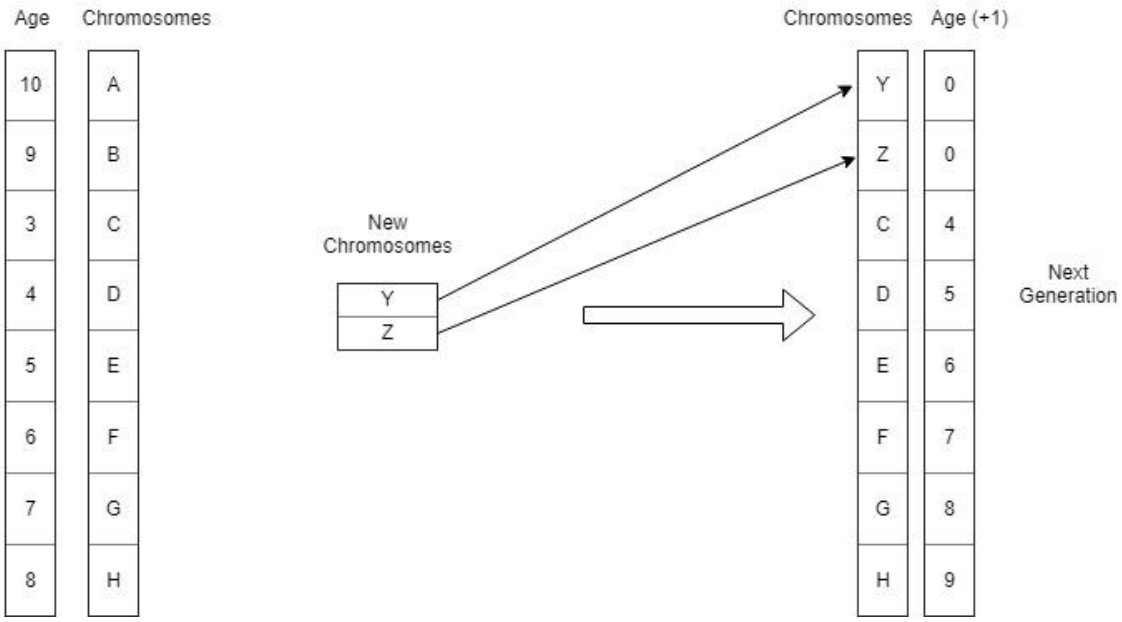

Figure 3.17: Age Based Selection

3.12 TERMINATION CONDITION

Termination condition is the last part of GA. The generation creation loop finishes according to this condition. These conditions may be applied as termination condition to a GA ;

- If there will be no further improvements after a point of iterations.
- If objective function hit to a pre-defined value.
- If pre-defined number of generations realized.

In our work, we used a pre-defined maximum number of generations as a termination condition. We also want to mention that termination condition is highly problem specific and it must be taken into account according to given problem.

4. ANALYSIS DETAILS

In this thesis, we have applied Genetic Algorithm with its various different methods and parameters on Traveling Salesman Problem(TSP). Our analysis consists of three different and neighbor geographic regions of Turkey that has 32 different cities in total. Theoretically and also in real life, all the cities have connected roads between them. We just measured theoretic air distance between all cities for brevity. Real road distances, real routes and traffic congestion also may be applied for further advanced researches.

There are 32 cities total in three different regions, Marmara (11 cities), Aegean(8 cities) and Black Sea(13 cities) regions. We use "Euclidean distances" between cities. Basically, put cities to a graph according to their x and y coordinates and here is the distance function between two cities :

$$city1 = (x_1, y_1),\ \ city2 = (x_2, y_2) \qquad (1)$$
$$d = \sqrt{(x_2 - x_1)^{\,2} + (y_2 - y_1)^2}$$

Using euclidean distance, we can get theoretical distances between cities, and also we can visualize cities and distances on a graph. Total distance of a chromosome is calculated as the following,

$$d_t = \sum_{i=0}^{n-1} d_i \qquad (2)$$
$$d_t : total\ distance\ of\ a\ chromosome$$

In our work, we analysed following parameters and methods during our trials ;

- Population size
- Maximum generation

- Crossover Method
- Crossover rate
- Mutation Method
- Mutation rate

We analysed the affects of each of these parameters to GA performance. Also we used objective function/fitness calculation as the distance between cities and total distance of a chromosome.

Table 4.1: Three Regions and Cities

Marmara Region		
City	X coord	Y Coord
Balikesir	100.0	475.0
Bilecik	300.0	400.0
Bursa	200.0	425.0
Canakkale	25.0	425.0
Edirne	50.0	187.0
Istanbul	200.0	125.0
Kirklareli	100.0	50.0
Kocaeli	225.0	175.0
Sakarya	325.0	225.0
Tekirdag	125.0	190.0
Yalova	200.0	200.0

Aegean Region		
City	X coord	Y Coord
Izmir	20.0	650.0
Manisa	100.0	600.0
Kutahya	220.0	550.0
Usak	200.0	650.0
Afyon	325.0	650.0
Aydin	100.0	750.0
Denizli	200.0	750.0
Mugla	50.0	850.0

Black Sea Region		
City	X coord	Y Coord
Eskisehir	350.0	525.0
Ankara	425.0	525.0
Cankiri	475.0	225.0
Kirikkale	480.0	525.0
Kirsehir	495.0	560.0
Yozgat	550.0	525.0
Sivas	650.0	525.0
Nevsehir	510.0	620.0
Konya	425.0	750.0
Aksaray	495.0	750.0
Kayseri	575.0	650.0
Nigde	510.0	775.0
Karaman	460.0	875.0

4.1 POPULATION INITIALIZATION

We chose ***random initialization*** method for population initialization in our work. Basically, identified the population size and we randomly chose different cities from the list and put randomly in a chromosome. The order is not important because Traveling salesman problem is bi-directional and direction of a route from a city to another is not important. There are 32 cities in a chromosome in different order(solutions). Randomness is provided by java.util.Random library of Java language.

4.2. FITNESS FUNCTION

Our fitness function is the same as objective function, total distance of a chromosome. To perform elitism and to find best chromosome among all the population, we try to find the best chromosome to compare total chromosome distance (total distance of chosen route). Figure 4.1 shows a small part the population of four chromosomes and its fitness values(total distance).

```
[ Kayseri Edirne Izmir Yozgat Afyon Bilecik Istanbul Karaman
Yalova Kirikkale Aksaray Usak Sakarya Mugla Kirklareli Manisa
Eskisehir Denizli Tekirdag Balikesir Kirsehir Kocaeli Kutahya Nigde
Sivas Aydin Nevsehir Konya Cankiri Bursa Ankara Canakkale ]
Distance: 14055

[ Mugla Ankara Manisa Kocaeli Nigde Aksaray Eskisehir Izmir Afyon Edirne
Denizli Istanbul Yozgat Nevsehir Yalova Konya Canakkale Sivas Sakarya
Bursa Tekirdag Kirklareli Aydin Kayseri Kutahya Kirsehir Balikesir
Bilecik Usak Karaman Kirikkale Cankiri ]
Distance: 13070


[ Kayseri Bursa Konya Eskisehir Kirikkale Canakkale Yozgat Kirklareli
Istanbul Cankiri Denizli Kutahya Sivas Edirne Nevsehir Ankara Balikesir
Nigde Sakarya Kocaeli Aksaray Usak Yalova Tekirdag Izmir Aydin Kirsehir
Karaman Afyon Bilecik Manisa Mugla ]
Distance: 11913


[ Aksaray Denizli Bilecik Usak Afyon Edirne Cankiri Kayseri Manisa
Karaman Kocaeli Kirklareli Izmir Kirikkale Nigde Nevsehir Bursa Kirsehir
Yozgat Canakkale Aydin Eskisehir Yalova Konya Balikesir Mugla Tekirdag
Istanbul Ankara Sakarya Sivas Kutahya ]
Distance: 12257
```

Figure 4.1: Fitness values of four example chromosomes

4.3 PARENT SELECTION

For the parent selection procedure, we use ***k-Way Tournament Selection***, as it is very efficient for our case and also profits from randomness, thus we can maintain the diversity among our population. We decided to choose k value for 3, as it chooses 3 chromosomes randomly and finally chooses a best one among them. We also give ***%20*** chance (relatively small chance) to maintain randomness, thus the diversity, that the best chromosome is not picked as a parent, instead, randomly choosen chromosome is

selected instead of the best one. We also apply elitism using 1 chromosome, that means, for each generation, only 1 best chromosome is selected for propagating directly to the new generation. Figure 4.2 shows our real example parent selection process.

Figure 4.2 : Parent Selection with Possible Real Chromosomes

4.4 APPLIED CROSSOVER METHODS

In this work, we implemented and analysed 3 main crossover methods ;

- One Point Crossover
- Two Point Crossover
- Uniform Crossover

After the parent selection, we decide to realize for a crossover between these parent or not according to crossover probability p_c. p_c is determined while program initialization. (e.g we choose a random number and see if this number is below p_c, if we determine p_c = 0.9, that means, the crossover operator will take place by %90 chance between parents, if chosen random number is above 0.9, then no crossover is implemented for this iteration.) Figure 4.3 shows our crossover probability implementation using pre-defined crossover rate.

$$\text{if } x \leq p_c \text{ , realize crossover} \qquad (2)$$
$$\text{if } x > p_c \text{ , no crossover}$$

```
boolean doCrossover = (random.nextDouble() <= crossoverRate);
if (doCrossover) {
    ArrayList<Chromosome> children = crossover(p1, p2);
    p1 = children.get(0);
    p2 = children.get(1);
}
```

Figure 4.3 : Crossover Probability Implementation

In our implementation, one point crossover and two point crossover methods are implemented as the algorithm presented in previous chapter, but for the uniform crossover method, we use bitmask that generates an array of a specified size with randomly places 1s and 0s. Then we decide which genes to replace according to that bitmask as, e.g. Example: child 1 has all the same cities as parent 1 at the indexes where the bit-mask is 1 and the same process is applied to child 2.

4.5 APPLIED MUTATION METHODS

Mutation methods are very important as they are used to maintain diversity. In our work, we chose very little chances of mutation rate to keep balance between keeping diversity and randomness. Primarly we have given %4 (mutation rate : 0.04) chance of mutation rate to provide randomness, and we change this mutation rate in time to analyse its effects. We used following mutation methods as mutation operators ;

- Insertion Mutation
- Swap Mutation (in our implementation we named it ***reciprocal mutation***)
- Scramble Mutation

After appliying crossover operators, we apply chosen(pre-defined) mutation technique according to predefined mutation probability p_c , as p_c is significantly small for not converging to random search. We increase this probability in time to analyse its effects on the performance of GA. Figure 4.4 shows the implementation which how we provide probability, we do the same process for both of chosen chromosomes according to two distinct random probabilities ;

```
boolean doMutate1 = (random.nextDouble() <= mutationRate);
boolean doMutate2 = (random.nextDouble() <= mutationRate);
if (doMutate1)
    p1 = mutate(p1);
if (doMutate2)
    p2 = mutate(p2);
```

Figure 4.4 : Mutation Probability Implementation

After all the crossover and mutation process is completed, if there is any room left, that means, if the next generation list contains less elements than population size, then we simply fill the empty places with random chromosomes in the last generation using k-way tournament selection, so population size remains the same.

4.6 TERMINATION CONDITION

In this work, we chose the termination condition as a pre-defined maximum generation number. Simply, if generation number hits the pre-defined value, then termination condition hit. We modified this pre-defined max. generation value in to to analyse its effects on the performance. We do not perform any other special conditions as early finishing with no further improvements etc. These kind of implementations are left for the future researches for brevity. Figure 4.5 shows the implementation of termination condition, we iterate over an array that the termination condition is ***maxGen***.

```
for (int i = 0; i < maxGen; i++) {
    population = createNextGeneration();

    Chromosome mostFit = population.getMostFit();
    if (!mostFit.equals(mostFitLast)) {
        if (draw) {
            win.draw(mostFit);
        }
    }
    mostFitLast = mostFit;
    averageDistanceOfEachGeneration.add(population.getAverageDistance());
    areaUnderAverageDistances += population.getAverageDistance();
    bestDistanceOfEachGeneration.add(population.getMostFit().getDistance());
    areaUnderBestDistances += population.getMostFit().getDistance();
}
```

Figure 4.5 : Termination Condition Implementation

5. RESULTS

In this work, we analysed the best distances and average of these best distances of each iteration according to the given parameters. Then we compared each parameters' effects to the final results. We created a for loop and run genetic algorithm for pre-defined parameters. e.g.

- Population size : range from 10 to 500 and increased by 30 ,
- Max. Gen : range from 10 to 500 and increased by 30,
- Crossover method :
    1. One point Crossover
    2. Two point Crossover
    3. Uniform Crossover
- Mutation method :
    1. Insertion Mutation
    2. Swap Mutation (Reciprocal Mutation)
    3. Scramble Mutation
- Crossover Rate
- Mutation Rate

In this work, we modified some parameters and we fixed other parameters to see effects of the changing parameter. First we run the GA according to given parameters in a loop, and calculated best distances and average distances in the end, we also calculate the average calculated distances of all crossover and mutation methods, so that we can compare all methods with averages and see how much optimization has been done compared to the averages. In the following section, you can find the result for several parameter effects:

5.1 POPULATION SIZE

5.1.1. First Trial

Population size parameter : Varies and increments from 10 to 500 by 30

MaxGen : ***100, fixed***

Crossover Methods : Varies, {Uniform, One Point, Two point}

Mutation Methods : Varies, {Insertion, Swap, Scramble}

Crossover Rate : 0.9

Mutation Rate : 0.04

Table 5.1 shows the yielded result ;

Table 5.1 : Results of GA with varying PopSize, Crossover and Mutation Methods

PopSize	Max Gen	Crossover	Mutation	Min. Dist.	Avg. Dist.	GA Dist. Avg.	% min dist. Opt.	% avg. dist. Opt.
10;500;(+30)	100	Uniform	Insertion	3781	4894	5204	27,34	5,96
10;500;(+30)	100	Uniform	Swap	3799	4639	5204	27,00	10,86
10;500;(+30)	100	Uniform	Scramble	3756	4889	5204	27,82	6,05
10;500;(+30)	100	One Point	Insertion	4153	5113	5204	20,20	1,75
10;500;(+30)	100	One Point	Swap	4305	5349	5204	17,28	-2,79
10;500;(+30)	100	One Point	Scramble	4404	5886	5204	15,37	-13,11
10;500;(+30)	100	Two Point	Insertion	4007	5007	5204	23,00	3,79
10;500;(+30)	100	Two Point	Swap	4087	5209	5204	21,46	-0,10
10;500;(+30)	100	Two Point	Scramble	4451	5858	5204	14,47	-12,57

As a result, when we increment the PopSize from 10 to 500, and the max PopSize is 500, and the MaxGen is defined to 100, all the crossover methods and mutation

methods yield good results but for finding the minimum distance, using uniform crossover and scramble mutation combined yielded slightly the best result (%27 optimization compared to average). But if we look at the average distances calculated, uniform crossover and swap mutation showed better results.(%10.86 optimization compared to average)

5.1.2 Second Trial

On this trial, we increment MaxGen to 300, here are the results;

Population size parameter : Varies and increments from 10 to 500
MaxGen : ***300, fixed***
Crossover Methods : Varies, {Uniform, One Point, Two point}
Mutation Methods : Varies, {Insertion, Swap, Scramble}
Crossover Rate : 0.9
Mutation Rate : 0.04
Figure 5.1 and Table 5.2 shows the yielded result ;

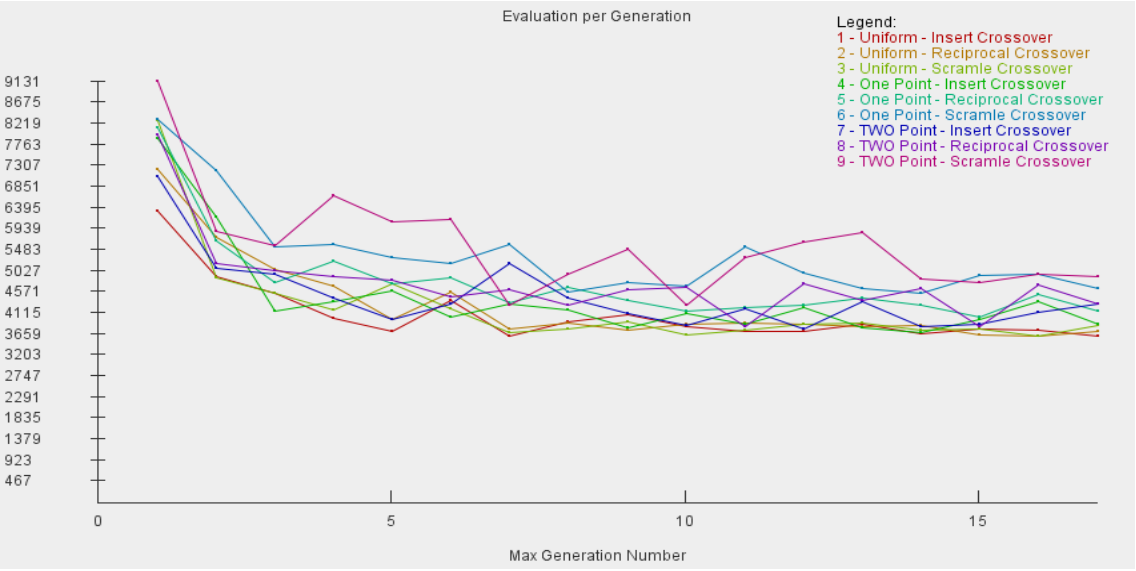

Figure 5.1 : Results on a Graph

Table 5.2 : Detailed Results

PopSize	Max Gen	Crossover	Mutation	Min. Dist.	Avg. Dist.	GA Dist. Avg.	% min dist. Opt.	% avg. dist. Opt.
10;500;(+30)	300	Uniform	Insertion	3618	4091	4675	22,61	12,49
10;500;(+30)	300	Uniform	Swap	3618	4300	4675	22,61	8,02
10;500;(+30)	300	Uniform	Scramble	3618	4264	4675	22,61	8,79
10;500;(+30)	300	One Point	Insertion	3693	4445	4675	21,01	4,92
10;500;(+30)	300	One Point	Swap	4042	4772	4675	13,54	-2,07
10;500;(+30)	300	One Point	Scramble	4555	5367	4675	2,57	-14,80
10;500;(+30)	300	Two Point	Insertion	3779	4473	4675	19,17	4,32
10;500;(+30)	300	Two Point	Swap	3823	4675	4675	18,22	0,00
10;500;(+30)	300	Two Point	Scramble	4287	5588	4675	8,30	-19,53

As a result, when we increment the PopSize from 10 to 500, and the max PopSize is 500, and the MaxGen is defined to 300, all the crossover methods and mutation methods yield better results but for finding the minimum distance, this time, GA distance average is also decreased from 5204 to 4675, that means, all the methods yielded better results than the previous trial. This time, 3 different methods combined yielded slightly the best result (%22,61 optimization compared to average). But if we look at the average distances calculated, uniform crossover and ***insertion*** mutation showed better results.(%12.49 optimization compared to average). For the final decision, incrementing MaxGen from 100 to 300 yields better results, as a conclusion we can say that incrementing max generation number with popsize also increases genetic algorithm ***performance*** and the ***optimization***.

5.2 MAXIMUM GENERATION SIZE

5.2.1. First Trial

Population size parameter : 100 , fixed

MaxGen : Varies and increments from 10 to 500 by 30

Crossover Methods : Varies, {Uniform, One Point, Two point}

Mutation Methods : Varies, {Insertion, Swap, Scramble}

Crossover Rate : 0.9

Mutation Rate : 0.04

Figure 5.2 and Table 5.3 shows the yielded result ;

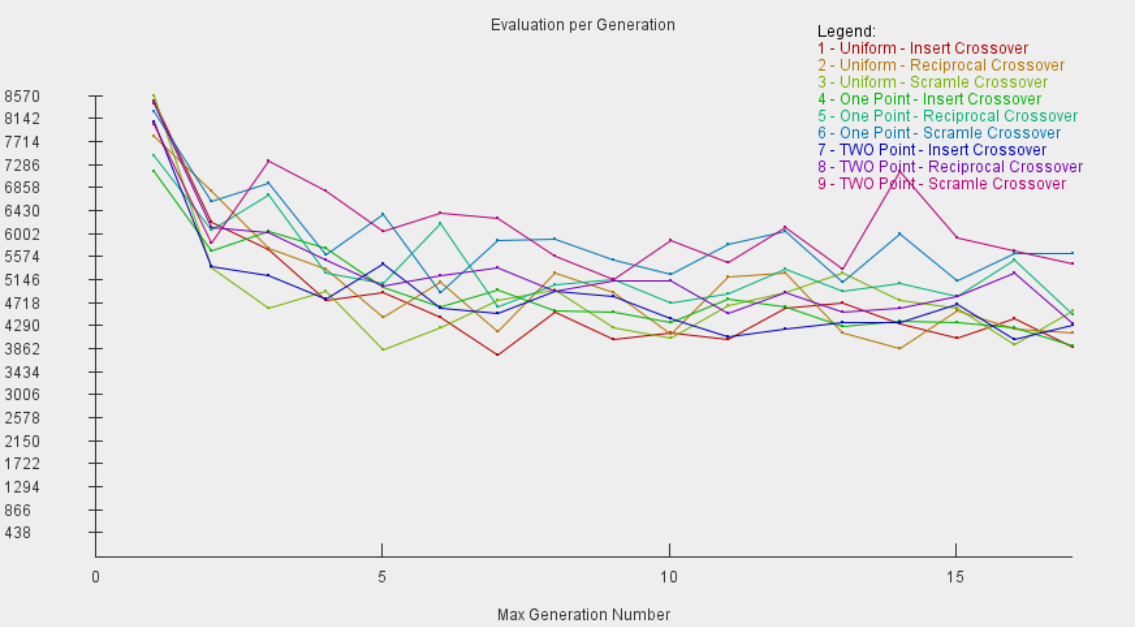

Figure 5.2 : Results on a Graph

Table 5.3 : Detailed Results

PopSize	MaxGen	Crossover	Mutation	Min. Dist.	Avg. Dist.	GA Dist. Avg.	% min dist. Opt.	% avg. dist. Opt.
100	10;500; (+30)	Uniform	Insertion	3756	4786	5250	28,46	8,84
100	10;500; (+30)	Uniform	Swap	3877	5027	5250	26,15	4,25
100	10;500; (+30)	Uniform	Scramble	3860	4864	5250	26,48	7,35
100	10;500; (+30)	One Point	Insertion	3943	4919	5250	24,90	6,30
100	10;500; (+30)	One Point	Swap	4520	5399	5250	13,90	-2,84
100	10;500; (+30)	One Point	Scramble	4919	5935	5250	6,30	-13,05
100	10;500; (+30)	Two Point	Insertion	4050	4860	5250	22,86	7,43
100	10;500; (+30)	Two Point	Swap	4335	5306	5250	17,43	-1,07
100	10;500; (+30)	Two Point	Scramble	5175	6162	5250	1,43	-17,37

As a result, when we increment the MaxSize from 10 to 500, and the MaxSize is 500, and the PopSize is defined to 100, all the crossover methods and mutation methods yield good results but for finding the minimum distance, this time, GA distance average is again increased from 4675 to 5250 that means, all the methods yielded worst results than the last trial. This time, Uniform crossover and Insertion mutations combined yielded slightly the best result (%28,46 optimization compared to average). As if we look at the average distances calculated, also the same combination showed better results.(%8,84 optimization compared to average). For the final decision, fxing popSize to 100 and varying maxGen is yielded slightly worst results than the previous trials in terms of average distance but this time, Uniform crossover and Insertion mutation combined is the winner for better optimizations among other methods. As a conclusion we can say that with lower popSize with increasing MaxGen, using Uniform crossover

and Insertion mutation combination is better for genetic algorithm ***performance*** and the ***optimization***.

5.2.2 Second Trial

Population size parameter : 300 , fixed

MaxGen : Varies and increments from 10 to 500 by 30

Crossover Methods : Varies, {Uniform, One Point, Two point}

Mutation Methods : Varies, {Insertion, Swap, Scramble}

Crossover Rate : 0.9

Mutation Rate : 0.04

Figure 5.3 and Table 5.4 shows the yielded result ;

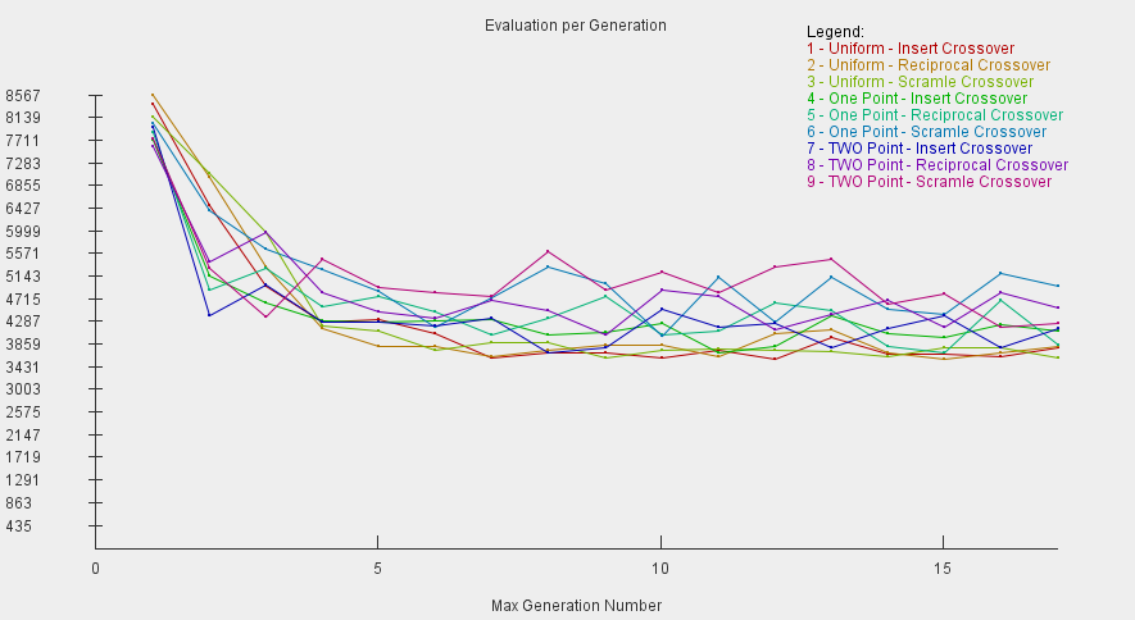

Figure 5.3 : Results on a Graph

Table 5.4 : Detailed Results

PopSize	MaxGen	Crossover	Mutation	Min. Dist.	Avg. Dist.	GA Dist. Avg.	% min dist. Opt.	% avg. dist. Opt.
300	10;500; (+30)	Uniform	Insertion	3602	4320	4636	22,30	6,82
300	10;500; (+30)	Uniform	Swap	3602	4388	4636	22,30	5,35
300	10;500; (+30)	Uniform	Scramble	3618	4396	4636	21,96	5,18
300	10;500; (+30)	One Point	Insertion	3725	4453	4636	19,65	3,95
300	10;500; (+30)	One Point	Swap	3708	4623	4636	20,02	0,28
300	10;500; (+30)	One Point	Scramble	4033	5139	4636	13,01	-10,85
300	10;500; (+30)	Two Point	Insertion	3725	4440	4636	19,65	4,23
300	10;500; (+30)	Two Point	Swap	4065	4859	4636	12,32	-4,81
300	10;500; (+30)	Two Point	Scramble	4196	5110	4636	9,49	-10,22

As a result, when we increment the MaxSize from 10 to 500, and the MaxSize is 500, and the PopSize is defined to 300 fixed, all the crossover methods and mutation methods yield better results than the previous trials, and also for the first time we could reach to the best minimum distance (3602, we found this value after hundreds of trials), for finding the minimum distance, this time, GA distance average is decreased from 5250 to 4636 that means, all the methods yielded better results than the last trial and previous trials. This time, Uniform crossover and Insertion mutations combined and Uniform crossover and Swap mutations combined yielded slightly the best results (%22,30 optimization compared to average). As if we look at the average distances calculated, Uniform crossover and Insertion mutation combination showed better results.(%6,82 optimization compared to average). For the final decision, fxing popSize to 300 and varying maxGen is yielded slightly better results than the previous trials in terms of average distance, so we can conclude that increasing maxGen and deciding

bigger popSize value also increases GA's overall performance. Also uniform crossover and Insertion mutation combined is again the winner for better optimizations among other methods. As a conclusion we can say that with higher popSize with increasing MaxGen, using Uniform crossover and Insertion mutation combination is better for genetic algorithm ***performance*** and the ***optimization***.

5.3 CROSSOVER RATE

Population size parameter : 100 , fixed
MaxGen : 100, fixed
Crossover Methods : Varies, {Uniform, One Point, Two point}
Mutation Methods : Varies, {Insertion, Swap, Scramble}
Crossover Rate : Varies and increments from 0,01 to 1 by 0.01
Mutation Rate : 0.04, fixed

For this analysis, we do not compared each methods, instead, we focused on general behaviour of all the methods and their overall reaction to crossover rate increase. We also chose relatively lower population size and maxGen to minimize popsize and maxgen effects on the algorithm performance so that we can see the effect of crossover rate better.
Figure 5.4 shows the results for behaviors of all methods to crossover rate increase, mutation rate fixed.

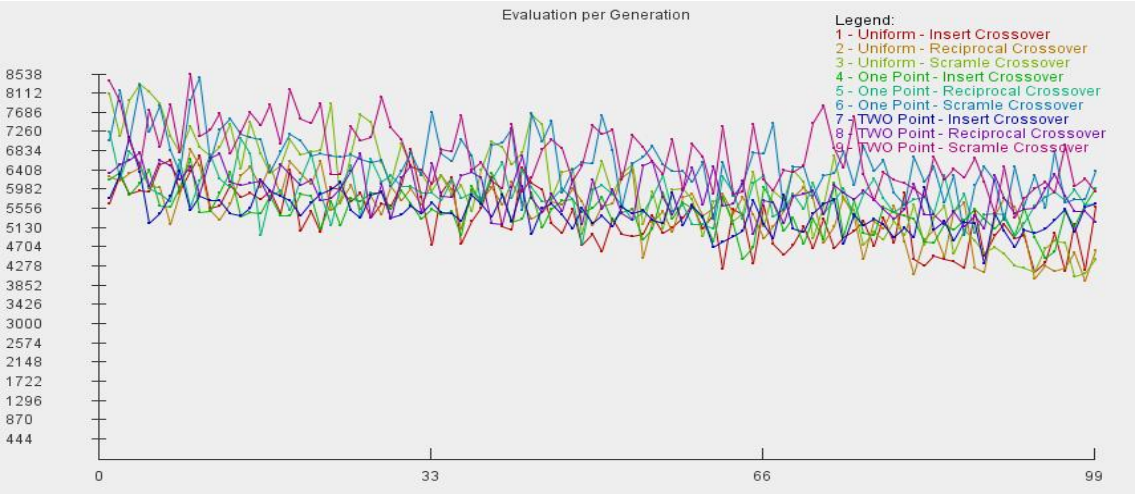

Figure 5.4 : General Behaviour of the Methods to Crossover Rate Increase

As you can see on the graph that while the crossover rate increases when the mutation rate fixed to 0.04, all the methods' performance gets better and yields better minimum distances, so as a conclusion, its better to choose a crossover rate close enough to 1 yields better performance in terms of optimization if processing time is not significantly important.

5.4 MUTATION RATE

Population size parameter : 100 , fixed
MaxGen : 100, fixed
Crossover Methods : Varies, {Uniform, One Point, Two point}
Mutation Methods : Varies, {Insertion, Swap, Scramble}
Crossover Rate : 0.9
Mutation Rate : Varies and increments from 0,001 to 1 by 0.002

Now, we know that higher crossover rate yields better results, then we analyse the mutation rate effect on the performance of the GA while crossover rate is fixed. As the

previous trial, this time, we fixed crossover rate and selected it as 0.9 because in our trials, higher crossover rate but not selecting it as 1 yields better results to see the effect of mutation rate. Also please note that calculation time of the process is not taken into consideration on this analysis as we do not have heavy load of input (32 cities in a chromosome).

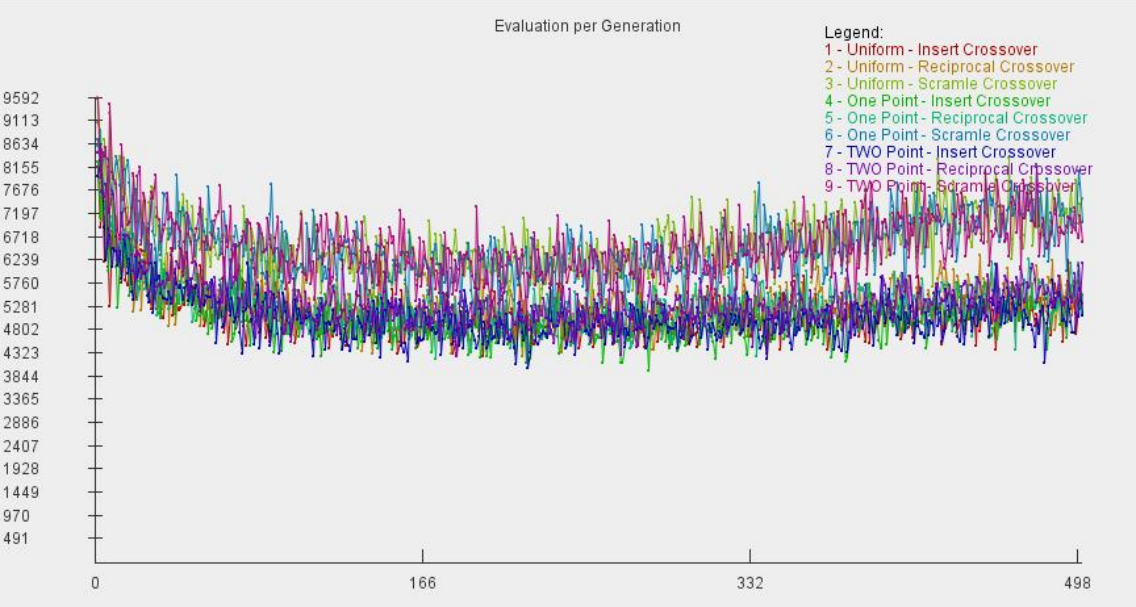

Figure 5.5 : General Behaviour of the Methods to Mutation Rate Increase

As you can see on the graph that while the mutation rate increases when the crossover rate fixed to 0.9, all the methods' performance gets better and yields better minimum distances around iterations between 150-250 and then minimum distances found are slightly increasing again, so as a conclusion, its better to choose a mutation rate between 0.03 to 0.05 gets better performance in terms of optimization if processing time is not significantly important. We also notified that, when the iteration gets higher (rate increase is small) process time also increases significantly.

6. FINAL RESULT & CONCLUSION

Final results show that in terms of both minimum distances and average distances found of the several crossover and mutation methods combined, nearly always, ***uniform crossover and insertion mutation method*** selection yielded best results, both for lower and higher popsize and maxGen.

On the other hand, our results show that, for our analysis (32 cities), selection of higher popSize and maxGen values also yielded better results but increasing popsize and maxgen parameters too high caused GA processing time to increase significant amount of times. Also for example, selecting popsize and maxGen parameters to higher than ***500***, did not give significant performance increase comparing to significant process time increase.

For the crossover rate and mutation rate analysis, we show that, when the other parameters are fixed, for crossover rate it is better to choose higher and closer values to 1 yields better results, ideally we found that selecting a value around ***0.85 - 0.95*** yielded best results for our case. On the other hand, for the mutation rate, as we mentioned in the previous chapters, increasing mutation rate does not yield better results after some point, the search is getting similar to random search so decreasing optimization of the GA. In our case, ideally, selecting mutation rate between ***0.03 - 0.05*** yielded best results.

REFERENCES

Allen, Franklin & Karjalainen, Risto. (1993). Using Genetic Algorithms to Find Technical Trading Rules. *Journal of Financial Economics*. 51. 10.1016/S0304-405X(98)00052-X.

Bethke, A.D., Genetic algorithms as function optimizers, University of Michigan, Ann Arbor, MI, 1980

BOYABATLI, Onur and SABUNCUOGLU, Ihsan. Parameter Selection in Genetic Algorithms. (2004). *Journal of Systemics, Cybernetics and Informatics*. 4, (2), 78-83

Chong, Edwin & Zak, S.H.. (1996). An Introduction to Optimization, third edn, John Wiley&Sons New Jersey

Grefenstette, John & Gopal, Rajeev & J. Rosmaita, Brian & Van Gucht, Dirk. (1985). *Genetic Algorithms for the Traveling Salesman Problem.*

HAUPT, L., Randy & Haupt, Sue. (1998). Practical Genetic Algorithms. 10.1002/0471671746.

Kim, Yongseog & Street, Nick & Russell, Gary. (2003). Customer Targeting: A Neural Network Approach Guided by Genetic Algorithms, *Management Science*, Vol. 51, No. 2, February 2005, pp. 264–276

Lin, L., Cao, L., and Zhang, C., Genetic Algorithms for Robust Optimization in Financial Application, *Proc. of the Fourth IASTED Inter. Conf. of Computational Intelligence*, pp. 387-391, 2005

Lin, li & Cao, Longbing & Wang, Jiaqi & Zhang, Chengqi. (2004). *The Applications of Genetic Algorithms in Stock Market Data Mining Optimisation*. 10.

Michalewicz, Zbigniew (1992), Genetic Algorithms + Data Structures = Evolution Program, first edn, Springer Newyork

Mills, Kevin & Filliben, James & L Haines, A. (2014). *Determining Relative Importance and Effective Settings for Genetic Algorithm Control Parameters. Evolutionary computation*. 23. 10.1162/EVCO_a_00137.

Parmal, Varun & Banerjee, Snigdha. (2017). Maximizing Sales of a Finance Company: A Genetic Algorithm Approach for Customer Identification. *Pacific Business Review*

International Volume 9 Issue 10, April 2017. 9. 49-54.

Potvin, Jean-Yves. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*. 63. 337-370. 10.1007/BF02125403.

Rexhepi, A., Maxhuni, A., Dika, A., Analysis of the impact of parameters values on the Genetic Algorithm for TSP, *IJCSI International Journal of Computer Science Issues*, Vol. 10, Issue 1, No 3, January 2013