



**T.C.**  
**SELÇUK ÜNİVERSİTESİ**  
**FEN BİLİMLERİ ENSTİTÜSÜ**

**Comparison Of Five Optimization Algorithms  
Based On Biological Inspiration For Travelling  
Salesman Problem**

**Omar Mohammed Ahmed Ahmed**

**YÜKSEK LİSANS TEZİ**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Eylül-2018**  
**KONYA**

## TEZ KABUL VE ONAYI

Omar Mohammed Ahmed Ahmed tarafından hazırlanan “Comparison Of Five Optimization Algorithms Based On Biological Inspiration For Travelling Salesman Problem ” adlı tez çalışması 04/10/2018 tarihinde aşağıdaki jüri tarafından oy birliği / ~~oy çokluğu~~ ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği** Anabilim Dalı’nda YÜKSEK LİSANS/~~DOKTORA TEZİ~~ olarak kabul edilmiştir.

### Jüri Üyeleri

### İmza

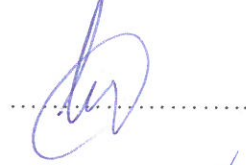
#### Başkan

Doç.Dr. Halife KODAZ



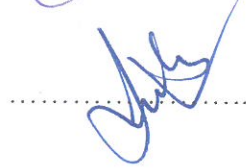
#### Danışman

Dr. Öğr. Üyesi Humar KAHRAMANLI



#### Üye

Doç.Dr. Gülay TEZEL



Yukarıdaki sonucu onaylıyorum.

Prof. Dr. Mustafa YILMAZ  
FBE Müdürü

Bu tez çalışması ..... tarafından ..... nolu proje ile desteklenmiştir.

## TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

## DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

İmza



Omar Mohammed Ahmed

Tarih:

## ÖZET

### YÜKSEK LİSANS

#### Gezgin Satıcı Problemi İçin Biyolojik Esinlemeye Dayalı Beş Algoritmanın Karşılaştırması

Omar Mohammed Ahmed Ahmed

Selçuk Üniversitesi Fen Bilimleri Enstitüsü  
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Humar KAHRAMANLI  
2018, 59 Sayfa

Jüri

Dr. Öğr. Üyesi Humar KAHRAMANLI  
Doç. Dr. Halife KODAZ  
Doç. Dr. Gülay TEZEL

1900'lü yıllardan beri, gezgin satıcı problemi (TSP), NP-hard optimizasyon problemlerine ait olduğu için en çok çalışılan optimizasyon problemlerinden biri olmuştur. TSP'nin ana fikri, Hamilton döngüsünün yaratılması için her şehir (düğüm) sadece bir kez ziyaret edilerek toplam mesafeyi en aza indirilmesidir. Başka bir deyişle, gezgin satıcı müşteri siparişlerini ilk şehir (düğüm) ile başlatarak ve her müşteriyi sadece bir kez ziyaret ederek ve başlayacak olan düğüme dönerek teslim ettiğinde, tüm bu süreç, seyahat edilen toplam mesafenin minimum olmasını sağlar. TSP'lerin klasik matematiksel yöntemler kullanılarak çözülmesi zordur. Günümüzde bile TSP problemlerini bu yöntemlerle çözen bilgisayarlar çok zaman almaktadır. Bu nedenle, gezgin satıcı problemi için birçok verimli optimizasyon algoritmaları, her zaman akademik önerilere odaklanmıştır. TSP'nin çoğu, gerçek zamanlı olarak tatmin edici çözümler sağlayan meta-sezgisel yöntemler ile çözülmüştür. Meta-sezgisel algoritmaları, hayvanların ve böceklerin karıncalar, arılar, balıklar, kuş sürüleri ve memeliler gibi bir çok çeşitli davranış yasalarının ilhamından icat edildi ve geliştirildi.

Bu tez beş meta-sezgisel yöntemine odaklanmaktadır: Gri Kurt Optimizasyonu, Balina Optimizasyon Algoritması, Tavuk Sürüsü Optimizasyonu, Karga Arama Algoritması ve Parçacık Sürü Optimizasyonu. Uygulama problemi TSPLIB'den seçildi. Bu tez aynı zamanda daha basit bir algoritmanın bile iyi bir çözüme ulaşabileceğini göstermektedir. TSP'yi çözmek veya meta-sezgisel çözüme başlamak için birincil algoritma olarak önerilebilecek olan muhtemelen en iyi sonucu üretecek yöntemler Balina Optimizasyon Algoritması ve Gri Kurt Optimizasyonudur. Bu tür çalışmaların temel amacı, büyük ölçekli TSP'yi uygun zamanda ve diğer birçok gerçek yaşam problemini çözmek için kullanılacak etkin ve etkili optimizasyon algoritmalarının geliştirilmesidir.

**Anahtar Kelimeler:** Gezgin Satıcı Problemi, Meta-Sezgisel Optimizasyon, Gri Kurt Optimizasyonu, Balina Optimizasyon Algoritması, Tavuk Sürüsü Optimizasyonu, Karga Arama Algoritması, Parçacık Sürü Optimizasyonu.

## **ABSTRACT**

## **MS THESIS**

### **Comparison Of Five Optimization Algorithms Based On Biological Inspiration For Travelling Salesman Problem Omar Mohammed Ahmed Ahmed**

### **THE DEGREE OF MASTER OF COMPUTER SCIENCE IN COMPUTER ENGINEERING**

**Advisor: Assist. Prof. Dr. Humar Kahramanlı**

**2018, 59 Pages**

**Jury**

**Assist. Prof. Dr. Humar KAHRAMANLI**

**Assoc. Prof. Dr. Halife KODAZ**

**Assoc. Prof. Gülay TEZEL**

Since the 1900s the travelling salesman problem (TSP) has been among the most widely studied optimization problems which belong to the NP-hard optimization problems. The main idea of TSP is that a Hamiltonian cycle will be created in a way that every city (node) will be visited once and only once leading the travelled total distance to be minimized. In other words the problem is when the salesman delivers customer orders by beginning with an initial city (node) then visiting every customer only once, and returning to the node that begin with, all that process should lead the travelled total distance to be the minimum cost tour. TSPs are difficult to be solved using classical mathematical methods. Even with nowadays computers solving TSP with these methods takes very plenty of time. Therefore, many efficient optimization algorithms for the TSP have been focused for academic proposes all the times. Most of the TSP are now solved by meta-heuristic methods, that provides a satisfactory solutions in real-time. Meta-heuristic algorithms were invented and developed from the inspiration of various behavior laws of animals and insects such as ants, bees, fish schools, bird flocks and mammals.

This thesis focuses on five meta-heuristic methods: Grey Wolf Optimizer, Whale Optimization Algorithm, Chicken Swarm Optimization, Crow Search Algorithm and Particle Swarm Optimization Algorithm. The problem for application was selected from TSPLIB. This thesis also shows that even a simpler algorithm can achieve quite good value of the solution. Probably the best implemented solutions were Whale Optimization Algorithm and Grey Wolf Optimizer which can be recommended as primary algorithm to solve the TSP or to start with the meta-heuristic solution. The main purpose of these kind of studies is the development of efficient and effective optimization algorithms that could be used to solve large scale TSP in appropriate time and many other real life problems.

**Keywords:** Travelling Salesman Problem, Meta-Heuristic Optimization, Grey Wolf Optimizer, Whale Optimization Algorithm, Chicken Swarm Optimization, Crow Search Algorithm, Particle Swarm Optimization.

## **Acknowledgements**

I would like to express my great gratitude to my supervisor, Assist. Prof. Dr. Humar Kahramanlı for her constant support and guidance. She introduced me to the field of optimization algorithms and she helped in the evolution of the ideas offered in this thesis by her constant feedback and supervision.

I would also like to thank my reader, Assist. Prof. Dr. Alaa Ali Hameed, for his valuable comments that have significantly improved this thesis. I would also like to thank my family and my friends for supporting me.

Omar Mohammed Ahmed Ahmed  
KONYA-2018

# CONTENTS

<b>ÖZET .....</b>	<b>I</b>
<b>ABSTRACT.....</b>	<b>II</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>II</b>
<b>CONTENTS.....</b>	<b>IV</b>
<b>SYMBOLS AND ABBREVIATIONS.....</b>	<b>VI</b>
<b>1. INTRODUCTION.....</b>	<b>1</b>
1.1. Purpose and Importance of Thesis.....	3
<b>2. LITERATURE REVIEW .....</b>	<b>4</b>
<b>3. MATERIALS AND METHODS .....</b>	<b>10</b>
3.1. Solution Techniques for TSP.....	10
3.1.1. Exact methods.....	10
3.1.1.1. Branch and bound method (B&B) .....	10
3.1.1.2. Cutting planes method.....	11
3.1.1.3. Branch and cut method.....	11
3.1.1.4. Dynamic programming method .....	11
3.1.1.5. Integer linear programming (ILP) method.....	11
3.1.2. Approximation methods.....	12
3.1.2.1. Heuristic optimization techniques.....	12
3.1.2.2. Local search algorithms .....	12
3.1.2.2.1. 2-Opt local search algorithm .....	12
3.1.2.2.2. 3-Opt local search algorithm .....	13
3.1.2.2.3. Lin-Kernighan Type Exchange .....	13
3.2 Optimization Algorithms .....	13
3.2.1 Grey wolf optimizer .....	16
3.2.2 Whale optimization algorithm .....	20
3.2.3 Particle swarm optimization .....	24
3.2.4 Chicken swarm optimization .....	29
3.2.5 Crow search algorithm.....	34
3.3 Pairwise Swap Mutation .....	40
<b>4. EXPERIMENTAL RESULTS AND DISCUSSION .....</b>	<b>41</b>
<b>5. CONCLUSION AND RECOMMENDATIONS.....</b>	<b>52</b>
5.1 Conclusion .....	52
5.2 Recommendations.....	53

**REFERENCES..... 54**

**CV..... 59**





## SYMBOLS AND ABBREVIATIONS

### Symbols

**D** : Distance

$f$  : Fitness value

**X** : Vector position of the wolf

$X_p$  : Vector position of the victim

$r_1, r_2$  : Randomly chosen vectors in [0, 1]

**A, C** : Coefficient vectors

**a** : Decreasing number from 2 to 0

**t** : Iterations

$X_\alpha$  : Alpha wolf

$X_\beta$  : Beta wolf

$X_\delta$  : Delta wolf

**X** : Vector position of the whale

$X^*$  : Position vector of the best solution

$D'$  : Distance between the victim (best solution) and the  $i^{th}$  whale

**b** : constant number

$l$  : Random number in [-1, 1]

$p$  : Random number in [0,1].

$X_{rand}$  : Random position vector (a random whale)

$w$  : Inertia weight

$c_1 c_2$  : Acceleration constants

$p_{id}$  : Personal best position

$p_{gd}$  : Global best position

$V_i$  :  $i$  particles velocity

$\alpha, \beta$  : Random number between 0 and 1

$randn(0, \sigma^2)$  : Gaussian distribution with mean 0 and standard deviation  $\sigma^2$

$\varepsilon$  : A constant number in the computer which is smallest number

$k$  : The index of the rooster's

$rand$  : Random number between [0, 1]

$x_{m,j}^t$  :  $i$ th Chick's mother position

$FL$  : Random number in [0,2].

$fl^{i,iter}$  : Flight length of crow  $i$

$m^{i,iter}$  : Hiding place position of crow  $i$

$AP^{j,iter}$  : Awareness probability of crow  $j$

## **Abbreviations**

**TSP** : Travelling salesman problem

**PSO** : Particle Swarm Optimization

**NI** : Nature-Inspired

**CSO** : Chicken Swarm Optimization

**GWO** : Grey Wolf Optimizer

**CSW** : Crow Search Algorithm

**WOA** : Whale Optimization Algorithm

**SO** : Swap Operator

**SS** : Swap Sequence

**GA** : Genetic Algorithm

**ACO** : Ant Colony Algorithm

**ABC** : Artificial Bee Colony

**MA** : Memetic Algorithm

**TSA** : Tabu-Search Algorithm

**SA** : Simulated Annealing

**FA** : Firefly Algorithm

**CSA** : Cuckoo Search Algorithm

**EA** : Evolutionary Algorithms

**ANN** : Artificial Neural Network

**AIS** : Artificial Immune Algorithm

**ILP** : Integer linear programming

**IWO** : Invasive Weed Optimization

**MDFA**: Multi-Population Discrete Firefly Algorithm

**HS** : Harmony Search Algorithm

**ABO** : African Buffalo Optimization

**BA** : Bat Algorithm

**BMA** : Biogeography Migration Algorithm

**CSA** : Clonal Selection Algorithm

**MISA** : Multi-objective Immune System Algorithm

**BCA** : B-Cell Algorithm

**BEA** : Bacterial Evolutionary Algorithm

**PeSOA**: Penguins Search Optimization Algorithm

**SSO** : Swallow Swarm Optimization

**CSO** : Cat Swarm Optimization

**HAS** : hunting search algorithm

**SFLA** : Shuffled Frog Leaping Algorithm

**SI** : Swarm Intelligence

**PB** : Physics-Based Algorithms

**GLSA** : Gravitational Local Search Algorithm

**CFO** : Central Force Optimization

**TLBO**: Teaching Learning Based Optimization

**LCA** : League Championship Algorithm

**SLC** : Soccer League Competition

## 1. INTRODUCTION

The Travelling Salesman Problem (TSP) is now among the complex and well-known NP-hard combinatorial optimization problems. It is easy to understand the TSP where it remains at the list of the one of the challenging problems of operational research. Its purpose is finding the shortest path for a salesman who must visit  $N$  cities. Solving TSP has both of practical importance and academic interest, and it is an important topic of active research.

There are huge numbers of methods and techniques have been invented to solve TSP. Some of them are Genetic Algorithms (GA) (Holland, 1992), Simulated Annealing (SA) (Bookstaber, 1997), Tabu-Search Algorithm (TSA) (Glover and Taillard, 1993), Ant Colony Optimization (ACO) Algorithm (Dorigo and Gambardella, 1997), Memetic Algorithm (MA) (Moscato, 1989), Bee Colony Optimization (BCO) Algorithm (Von Frisch, 1974), Firefly Algorithm (FA) (Yang, 2009), Cuckoo Search Algorithm (CSA) (Yang, 2010). In spite of classical algorithms such as TSA and SA are not that efficient to be used for solving optimization problems, Evolutionary Algorithms (EA) such as MA and GA gives appropriate solutions for complex optimization problems.

TSP can be explained as follow: Give the shortest path that covers all cities along. Let's assume that  $R = (N; S)$  be a graph where the  $N$  is a collection of vertices and  $S$  is a collection of edges. Let  $C = (C_{ij})$  be a cost (or distance) matrix related with  $S$ . In the TSP minimum distance loop (Hamiltonian loop or cycle) determination required, which is all the vertices are visited just one time. Assume that salesman already knows  $C_{ij}(i, j \in \{1, 2, 3, \dots, N\})$  which indicates the distance between the  $i$ th and  $j$ th cities. The salesman must select the route with the minimum travel distance. Besides it this tour must include all of the cities moreover each city must appear only one time. The salesman could begin his route from any city, while he must return to the city where he began his tour.

The need of quick find of satisfactory solutions to TSP has caused to the development of numerous methods such as meta-heuristics. Meta-heuristic algorithms have showed effective performance in solving a large set of optimization problems. Advantages of Meta-heuristic algorithms are more and better than classical methods such as flexibility and simplicity. Meta-

heuristic methods are generally easy to implement and proceed. In addition, these methods are very simple and flexible, and they are able to deal with many problems, both continuous and discrete moreover mixed.

Nowadays, the techniques which is using for TSP divides into two main groups: approximation algorithm, and exact methods which guarantees obtaining the optimal solution.

Approximation algorithms have the ability to obtain more accurate, therefore they are very appropriate to solve large-scale problems. These algorithms also divide into two groups: meta-heuristic optimization techniques and local search algorithms. Meta-heuristic optimization techniques search around the optimal solution. GA (Holland, 1992), SA (Bookstaber, 1997), ACO (Dorigo and Gambardella, 1997), PSO (Eberhart and Kennedy, 1995), Artificial Neural Network (ANN) (Liu and Xiu, 2007; Huang and Du, 2008; Han and et al., 2010), Marriage in Honey Bees Optimization Algorithm (Abbass, 2001) and Artificial Immune Algorithm (AIS) (Hunt and Cooke, 1996) are examples for heuristic optimization techniques. 2 - Opt (Croes, 1958), 3 - Opt (Lin, 1965), LK (Lin and Kernighan, 1973), LKH (Helsgaun, 2000) and Inver-over (Tao and Michalewicz, 1998) are the examples for the local search algorithms.

The second main category for solving TSP is exact methods which have the ability to obtain guarantee optimal solutions, but it leads to increasing in the problem's scale, the required time for solving exponentially increases. The common exact techniques include branch and bound method (Lawler and Wood, 1966), dynamic programming method (Bellman and Dreyfus, 1962).

In the past years, many researches could combine meta-heuristic algorithms with local search to develop a novel hybrid algorithm to solve TSP, such as LK and genetic operators (Merz and Freisleben, 1997), combined ACO with mutation strategy (Yang and et al., 2008), combined technique of a 2-Opt and GA (Samanlioglu and et al., 2008). These combined algorithms can get satisfactory solution in less iteration. In addition, the above mentioned heuristic, meta-heuristic algorithms and exact algorithms have been tested by number of developers on TSP successfully.

This thesis examines five nature-inspired (meta-heuristic) algorithms (wolf optimizer, whale optimization algorithm, chicken swarm optimization, crow search algorithm and particle swarm optimization) to calculate the solutions of TSP. Six benchmark problems were selected for the algorithms to test their performance, and the obtained solutions of the algorithms show that there

are differences in the performance of the algorithms. The rest sections of the thesis is ordered as follow: in section two detailed information about the TSP is given; section 3 gives brief explanation about applications, in section 4 simulations and result comparisons are presented with recommendations. In section 5 the work is concluded.

### **1.1 Purpose and Importance of Thesis**

The main purpose of this work is to find least distance value and time for TSP (Hamiltonian cycle). In this study we used five meta-heuristic algorithms (PSO, GWO, CSA, WOA and CSO) to perform it on TSP and compare the results with each other.

Furthermore, we add swap mutation to the algorithms to improve their performance and ability to obtain better results.

During the experimentation phase, six data sets for TSP are selected and best parameter's values for these algorithms adjusted to obtain the best results.

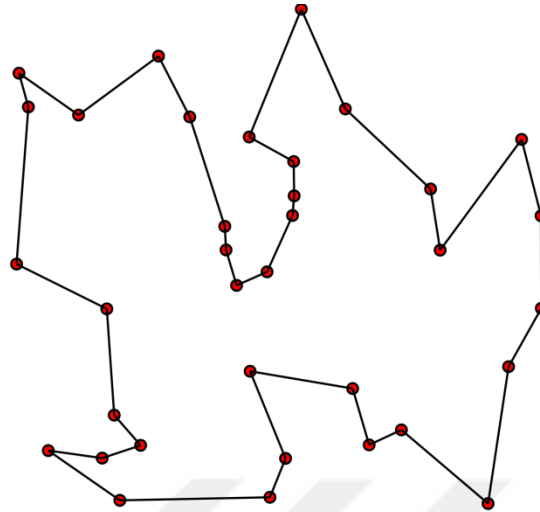
## 2. LITERATURE REVIEW

During the 1800s, some scientists started studying some mathematical problems related to the TSP, one of them was the mathematician William Rowan Hamilton, Irish physicist and the British mathematician Thomas Penyngton Kirkman (URL1; URL2) (Hamilton, 2000). After some years in Vienna in the 1930s the mathematician Menger (Menger, 1932) presented the problem in an academic conference. After some years in Princeton University Merrill Flood (URL3) and Hassler Whitney (URL4) used TSP in their studies and it looked like that the “Travelling Salesman Problem” name was firstly called by Whitney (Lawler, 1985). In spite of the fact in (Gutin and Punnen, 2006) that the stated work of (Menger, 1932) declared that the TSP is studied as Messenger Problem, and it was the initial study of TSP that published at that time. After a while the earliest studies on TSP began such as (Mahalanobis, 1940; Robinson, 1949; Dantzig and et al., 1954; Flood, 1956).

The studies of Dantzig and et al., 1954 are considered as the first computational study on TSP in a way that a problem with 49 cities is used in the study and the authors used linear programming techniques to give the obtained solution. After that, for the comparison of different solution algorithms a huge number of datasets of the problem have been used and created. The used well-known datasets problems are available in (URL5).

In Figure 2.1 an example of a Hamiltonian cycle is showed. It is obvious in the obtained tour that every city (node) is visited only once and the tour would be a closed tour meaning that the salesman completes the tour by returning to the city that started with.





**Figure 2.1.** An example of a TSP solution

This section gives an overview of previous work done on TSP. Previous work on TSP has been integrated into this literature overview for several reasons. Most of the algorithms used for solving TSP during the past years have been improved and adapted using (mutation, crossover, 3-Opt local search operator and etc.) and their results will be shown in Table 4.2.

In (Zhou and et al., 2015) they used Invasive Weed Optimization (IWO) algorithm to calculate its performance on TSP that is developed from a popular phenomenon in agriculture: colonization of invasive weeds. At the first step, positive integer is encoded by the weeds individuals, on the basis that changing the normal distribution of the IWO will not be completed, after that the fitness value will be calculated for the weeds individuals. At the second step, they used the 3-Opt local search operator to make improvements on the solution of TSP. At the final step, they used an improved complete 2-Opt (I2Opt) that is considered as a second local search operator to make improvements on the solution of TSP. They used 20 TSP benchmark datasets from TSPLIB such as (Att48, Eil51, Berlin52, St70, and KroA100) and the obtained results were near to optimal value with standard deviation below one.

In (Zhou and et al., 2014) they used FA to solve TSP which is inspired from the flashing behavior of firefly insects. They improved FA by that making it multi-population discrete firefly algorithm (MDFA) and adding k-opt algorithm to the process to solve TSP. They used 7 TSP

benchmark datasets to test the algorithm such as (Eil51, Berlin52 and St70) and the obtained results were near to the optimal value.

In the work (Bouzidi and Riffi, 2014) harmony search (HS) algorithm have been used to solve TSP. HS have been adapted and improved to solve TSP in a better way. There are three stages of adaption used in HS. The first stage is the initialization of the HMS, PAR, and HMCR settings. The second stage of the adaption is in a random way there will be initialization of the harmony memory (HM) of the HM solutions so that a Hamiltonian cycle of cities will be represented in each solution. The third stage of the adaption is starting to search for a solution that relies on the values of the parameters until stopping criteria is reached. More than 30 TSP benchmark datasets were used to calculate the performance of the algorithm and the obtained solutions were optimum in most of the datasets.

In the work (Odili and Mohmad Kahar, 2016) African Buffalo Optimization (ABO) have been used to solve TSP which is inspired from the behavior of African buffalos, that belong to a group of wild cows, these animals live in savannahs and in African forests. The ABO has the advantage to be used for complicated optimization problems such as TSP because it has simple steps. Buffalos have the ability to guarantee an outstanding exploitation and exploration of the territory that they live by using regular cooperation, communication, and using its previous personal experience as a good memory as well as taking in considers the herd's collective experience. More than 30 TSP benchmark datasets were used to calculate the performance of the algorithm and the obtained results were near to the optimum value in most of the datasets.

In the work (Osaba and et al., 2016) Bat Algorithm (BA) have been used to solve TSP which is inspired from the echolocation behavior or bio-sonar abilities of micro bats. Some improvements have been used on basic BA to solve TSP in a better way. The improvements were using two well-known local search operators for the movement of the bats. The first one is 2-Opt local search operator and the second one is 3-Opt local search operator. Also, they added to all the bats of the swarm a kind of intelligence. Meaning that, each bat in the swarm moves in a different way taking in considers its position and the position of best bat of the swarm. More

than 30 TSP benchmark datasets were used to calculate the performance of the algorithm and the obtained results were optimum in most of the datasets.

In the work (Mo and Xu, 2011) Biogeography migration algorithm (BMA) have been used to solve TSP which is inspired from the migration strategy of animals. Each individual at the population is taken as a “habitat” with a habitat suitability index that is like the fitness value of EAs, to calculate the individual. A superior obtained result is identical to an island with a high habitat suitability index, and a bad result is identical an island with a low habitat suitability index. High habitat suitability index results head to participate their features with low habitat suitability index results. Low habitat suitability index results take many new features from high habitat suitability index results. They used 9 TSP benchmark datasets to test the algorithm such as (KroA100, KroB100, KroC100, and KroD100) and the obtained results were near to the optimal value.

In the work (Pang and et al., 2015) Clonal Selection Algorithm (CSA) have been used to solve TSP that is related to the area study of AIS. This algorithm has a relation with different CSAs such as Multi-objective Immune System Algorithm (MISA), The B-Cell Algorithm (BCA), and the Artificial Immune Recognition System. Some improvements have been used on basic CSA to solve TSP in a better way. The improvements were using 2-Opt local search operator. They used 4 TSP benchmark datasets to test the algorithm (Eil51, Eil75, Eil101 and Berlin52) and the obtained results were near to the optimal value.

In the work (Meng and et al., 2016) ABC applied to calculate its performance in solving TSP. Some improvements have been used on basic ABC to be applied on TSP in a better way. The improvements were redefining the transforming mechanism and searching strategy of scout bees, leading bees and following bees according to discrete variables. The Leading bees use learning operator and 2-Opt operator to search the neighborhood and to make the convergence speed faster. To make the local refinement ability of the algorithm better the process of searching of following bees introduces tabu table for it. An exclusive operation is defined by Scouts bees to maintain the diversity of population. Five TSP benchmark datasets were used calculate the

performance of the algorithm and the obtained solutions were optimum in small datasets and near to the optimal value in big datasets.

In the work (Kóczy and et al., 2017) Bacterial Evolutionary Algorithm (BEA) have been used to solve TSP which is taken from the biological phenomenon of microbial evolution. Some improvements have been used on BEA to solve TSP in a better way. The improvements were using bacterial mutation optimizes, Loose Segment Mutation, Coherent Segment Mutation and Gene Transfer. Moreover they used two well-known local search operators in the algorithm. The first one is 2-Opt local search operator and the second one is 3-Opt local search operator. Fifteen TSP benchmark datasets were used to calculate the performance of the algorithm and the obtained solutions were optimum in most of the datasets.

In the work (Mzili and Riffi, 2015) Penguins Search Optimization Algorithm (PeSOA) have been used to solve TSP which is based on the hunting strategy of the Penguins. In this study PeSOA have been adapted depending on the hunting strategies of the Penguins. At the first phase parameters of PeSOA adapted with TSP, at the second phase the equation database system of PeSOA are rewrote to improve the penguins position find optimal solutions values. 30 TSP benchmark datasets were used to calculate the performance of the algorithm and the obtained solutions were optimum in all the datasets.

In the work (Bouzidi and Riffi, 2017) Swallow Swarm Optimization (SSO) have been used to solve TSP which is inspired from the intelligent behaviors of swallows. In this work SSO have been adapted to solve TSP in a better way. The algorithm was adapted by using operators and operations. Elementary operations were used that consists of (Addition operator, Subtraction operator, Multiplication operator). 10 TSP benchmark datasets were used to calculate the performance of the algorithm and the obtained solutions were optimum in most of the datasets.

In the work (Bouzidi and Riffi, 2013) Cat Swarm Optimization (CSO) have been used to solve TSP which is inspired from the natural behavior of cats. Some improvements have been used on Cat Swarm Optimization to solve TSP in a better way. The improvements were using different operators and operations that are performed in two different modes of this algorithm.

Firstly the mode of the search (SM) and secondly the mode of the trace (TM). The operations (Opposite of velocity, Addition, Subtraction and Multiplication) are applied on the position and the velocity of each member of the cats in the tracing mode. More than 10 TSP benchmark datasets were used to calculate the performance of algorithm and the obtained solutions were optimum in all the datasets.

In the work (Agharghor and Riffi, 2015) Hunting Search Algorithm (HSA) have been used to solve TSP which is inspired from the technique of hunting group of predator animals. In this work HSA has been adapted in a discrete case. Operations of the algorithm will be redefined into operations of permutation in the path of the visited cities of the TSP. 30 TSP benchmark datasets were used to calculate the performance of the algorithm and the obtained solutions were optimum in almost all the datasets.

In the work (Saud and et al., 2018) Shuffled Frog Leaping Algorithm (SFLA) have been used to solve TSP which is inspired from natural memetics. Two improvements have been used on SFLA in order to calculate its performance on TSP in a better way. The first improvement was adding cycle crossover to the algorithm and the second improvement was adding crossover with inversion mutation to the algorithm. But it seems that the improvements were not a good choice to solve TSP with that algorithm. Six TSP benchmark datasets were used to calculate the performance of the algorithm and the obtained solutions were too high from the optimum in all the datasets.

In the work (Kumbharana and Pandey, 2013) FA have been used to solve TSP that is taken from the rhythmic flashing behavior of Fireflies. In this work FA have been adapted to solve TSP in a better way. The algorithm was adapted by implementing functions `Initial_Solution()` and `Distance(xi, xj)` in the same method how it is represented in TSP. As well as, movements of the fireflies are redefined in another way. Six TSP benchmark datasets were used to calculate the performance of the algorithm and the obtained solutions were near optimum value in most of the datasets.

### 3. MATERIALS AND METHODS

#### 3.1 Solution Techniques For TSP

After the problem has been presented to the world, developers and scientists started studying TSP very extensively and they presented many solution techniques and methods which are proposed in the literature. Solving methods of TSP could be classified into two main methods exact methods which guarantee obtaining the optimal results and approximation algorithm (heuristic methods).

##### 3.1.1 Exact methods

Exact methods for solving TSP have the ability to find exact optimum solution value by enumerative search process. But on the other hand these methods for solving TSP are not that useful because when they are used to solve large datasets they require very big computational time. Exact methods have many algorithm types such as, branch and bound method, cutting planes algorithm, branch and cut method, dynamic programming and integer linear programming formulations.

**3.1.1.1 Branch and bound method (B&B).** This technique was presented by Land in 1960 which is considered as an exact algorithm that used to solve many optimization problems (Land and Doig, 1960). This method solves the problem to find the optimal solution value by a separating process called as “branching” the other stage is investigation stage called as “bounding”. When branching process works by dividing the problem into sub problems, then the bounding process chooses what branch to be completed.

In Branch and Bound Method, the solved problem itself is determined by using sub problems at the next stage bounds would be determined for each sub problem so bad result solutions would not be precede. The basic idea behind this method the bounds are used to help to compare old solutions with the new solutions meaning that stay away from searching around the solution that will not lead to the optimal solution value (Lawler and Wood, 1966).

**3.1.1.2 Cutting planes method.** This technique uses assumption to search a solution for the problem which is depending on finding integer values for objective function and variables. The method at the first stage obtains the lower bound of the problem by solving the linear programming relaxation (LP). After that a solution consists of source row will be chosen from LP relaxation. According to the chosen source row of the solution the cutting plane deletes a group of non-integer solutions. To solve LP problem a cutting plane will be added to the simplex tableau. After all that when the variables are all integer, means that the best solution value is reached (Winston and Goldberg, 2004)

**3.1.1.3 Branch and cut method.** This technique was invented from the combination of the branch and bound method and the cutting planes method. Depending on the solution result of the cutting planes method bounds will be determined in the branch and bound method; so it guarantee finding the best solution value in minimal space of feasible solutions (Mitchell, 2002) (Winston and Goldberg, 2004)

**3.1.1.4 Dynamic programming method.** This technique uses sequential steps to solve a problem. When a step is completed, the next step will start from using previous steps solution. When a small sub problems are completed, large sub problems are completed in correspond to the obtained solution values of the small sub problems (Bellman and Dreyfus, 2015).

**3.1.1.5 Integer linear programming (ILP) method.** This technique guarantees an exact solution value by definition of constraints, objective function and decision variables. In this technique, according to the characteristics of the used problem the structure of the variables, objective function and constraints can be modified and organized. In some cases ILP techniques will not useful in finding the optimal solution value of the problem. Thus, we can conclude that ILP techniques are better to be used in small size problems that do not require solutions in small time.

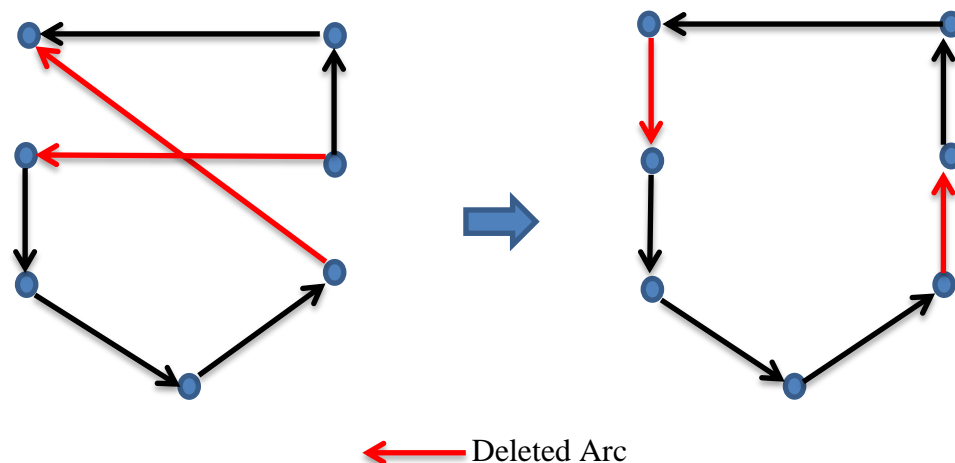
### 3.1.2 Approximation algorithms

Approximation algorithms are used to solve large-scale problems because they have the ability to obtain more accurate solution in appropriate time. These algorithms also classified into two main groups.

**3.1.2.1 Heuristic optimization techniques.** Heuristic optimization techniques try to reach around the optimal solution value. PSO, GA and ACO are examples for heuristic optimization techniques.

**3.1.2.2 Local search algorithms.** Local search algorithms have many methods that are used for solving TSP such as (2-Opt local search algorithm, 3-Opt local search algorithm and Lin-Kernighan Type Exchange).

**3.1.2.2.1 2-Opt local search algorithm.** This algorithm was presented by Croes in 1958 as a technique for solving TSP (Croes, 1958). The basic idea of this algorithm is that select two edges and then remove them and then reconnect them in another style to get a different Hamiltonian cycle that can obtain a superior solution. An example of 2-opt local search algorithm is shown in Figure 3.1.



**Figure 3.1.** An example of 2-opt local search movement.



**3.1.2.2.2 3-Opt local search algorithm.** This algorithm was presented by Bock in 1958 as a technique for solving TSP (Bock, 1958). The basic idea of this algorithm is that select three edges and then remove them and then reconnect them in another style to get a different Hamiltonian cycle that can obtain a superior solution. An example of 3-opt local search algorithm is shown in Figure 3.2.

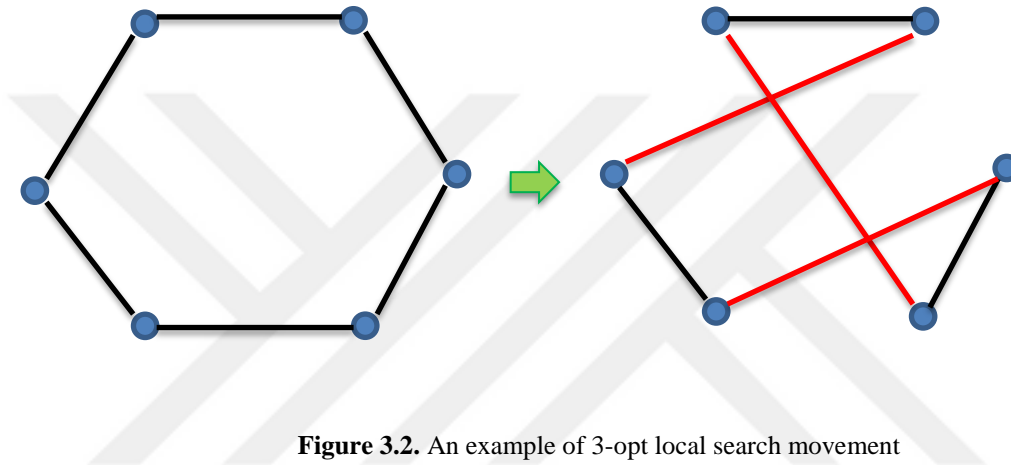


Figure 3.2. An example of 3-opt local search movement

**3.1.2.2.3 Lin-Kernighan type exchange.** This algorithm was presented by Lin in 1973 (Lin and Kernighan, 1973) as a technique for solving TSP. This method is different than 2-opt and 3-opt in choosing the number of nodes that will swap in between them where in this algorithm the number is specified dynamically. This technique depends on a search technique in a way that numbers of several transformations are applied. The applied transformations sometimes might not lead to a satisfied solution.

### 3.2 Optimization Algorithms

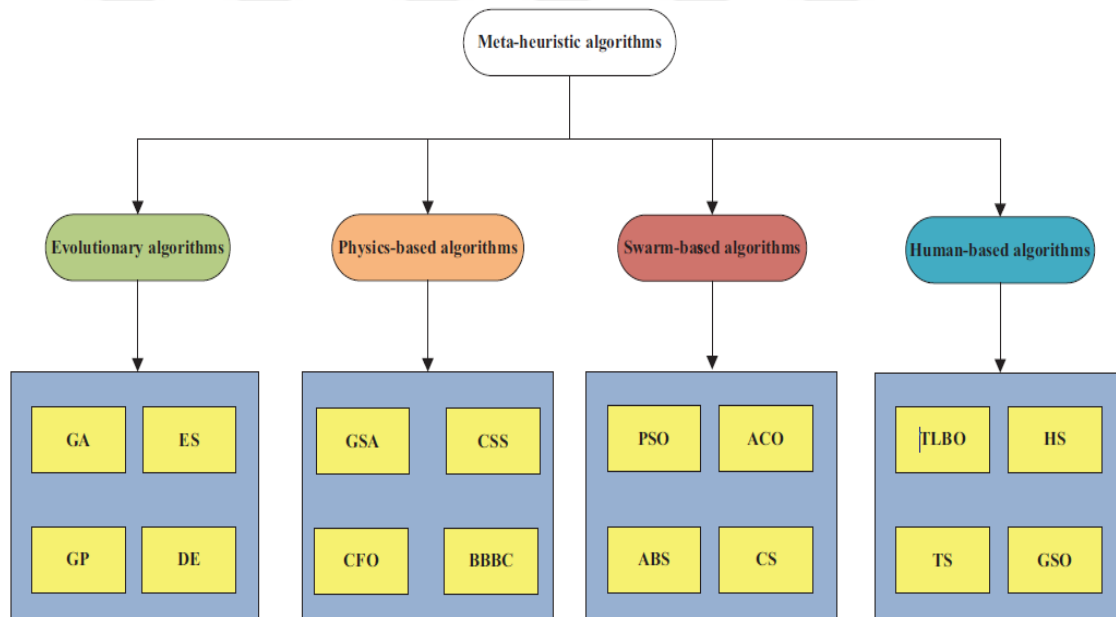
Optimization algorithms are used as a mechanism to find the global optimum value of a problem under certain circumstances. In our life there are many complex optimization problems that are raised in various scientific fields like engineering problems, business and economics problem and these problems seems difficult to be solved with a rational solution or time using classical mechanism (Li and et al., 2016).

In the nature there are various sources of concepts, principles and mechanisms that can be used to solve this kind of complex optimization problems by designing artificial computational

methods. In the past years, developers could design and invent various nature-inspired (NI) techniques to apply them on different optimization problems. These algorithms imitate a specific behaviors or phenomena in the nature such as Hunting search (Oftadeh and et al., 2010) taken from the behavior of group hunting of animals (dolphins, wolves, and lions), Monkey search (Mucherino and Seref, 2007) inspired from the life style of apes climbing trees searching for food, Dolphin echolocation (Kaveh and Farhoudi, 2013) inspired from the behavior of dolphins using echolocation to search for food. Comparing these algorithms with classical heuristic methods, it can be observed that this algorithms' performance are more efficient, particularly for optimizing the discrete complex, non-differentiable and multimodal optimization problems. Meantime, developers could use NIs widely in various scientific fields such as task scheduling, image processing, data mining applications, dynamic optimization, mechanical design problems, and some other engineering problems (Li and et al., 2016). Generally, NIs can be mainly divided into four groups: physics-based (PB) algorithms, swarm intelligence (SI) algorithms and evolutionary algorithms (EA) as shown in Figure 3.3.

- Evolutionary algorithms (EA): they are taken from the genetic and evolutionary behaviors of creatures in the nature such as GA.
- Swarm intelligence algorithms (SIs): these algorithms are also inspired from the intelligent behaviors of creatures in the nature. Most of the SIs follow genetic rules only and they always use every result in search space as an advantage to obtain superior solution values to the problem (Li and et al., 2016).
- Physics-based methods: these algorithms mimics physical rules in the universe such as Gravitational Local Search Algorithm (GLSA) (Webster and Bernhard, 2003) and Central Force Optimization (CFO) (Formato, 2007).
- Human-based algorithms: it's worth mentioning that there are algorithms inspired from the behavior of the human such as Teaching Learning Based Optimization(TLBO) (Rao and et al., 2011; Rao and et al., 2012), League Championship Algorithm (LCA) (Kashan, 2009; 2011) and Soccer League Competition (SLC) Algorithm (Moosavian and Roodsari, 2014a; 2014b)

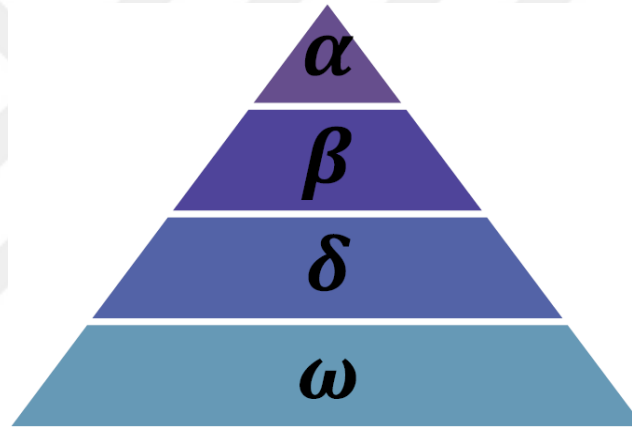
Among these algorithms swarm-based algorithms have the advantage over the others because they need fewer operators and they are easy to implement. There are common features in the population based meta-heuristic optimization algorithms without taking in consider their nature. In this kind of algorithms the searching operation can be sectioned into two main steps: exploration and exploitation (Olorunda and Engelbrecht, 2008; Lin and Gen, 2009). There must be operators in the optimizer to completely explore the search area: at that step, there should be randomizing in the movements as most as possible. After the exploration stage exploitation stage starts which is known as the operation of examining the promising area(s) of the search space in detail.



**Figure 3.3.** Classification of meta-heuristic algorithms

### 3.2.1 Grey Wolf Optimizer

GWO was presented by Seyedali Mirjalili, Seyed Mohammad Mirjalili and Andrew Lewis which is taken from grey wolves (*Canis lupus*) they belongs to Canidae family (Mirjalili and et al., 2014). Grey wolves are studied and considered at the highest degree of the food chain (apex predators). Grey wolves mostly choose to stay in groups of size 5–12 on average. Hunting strategy of grey wolves and the leadership hierarchy in nature is mimicked by the GWO algorithm. The grey wolves group consists of four types of wolves like alpha, beta, delta, and omega which are applied to simulate the leadership hierarchy. Figure 3.4 shows the social dominant hierarchy of the wolves which is very strict in the group.



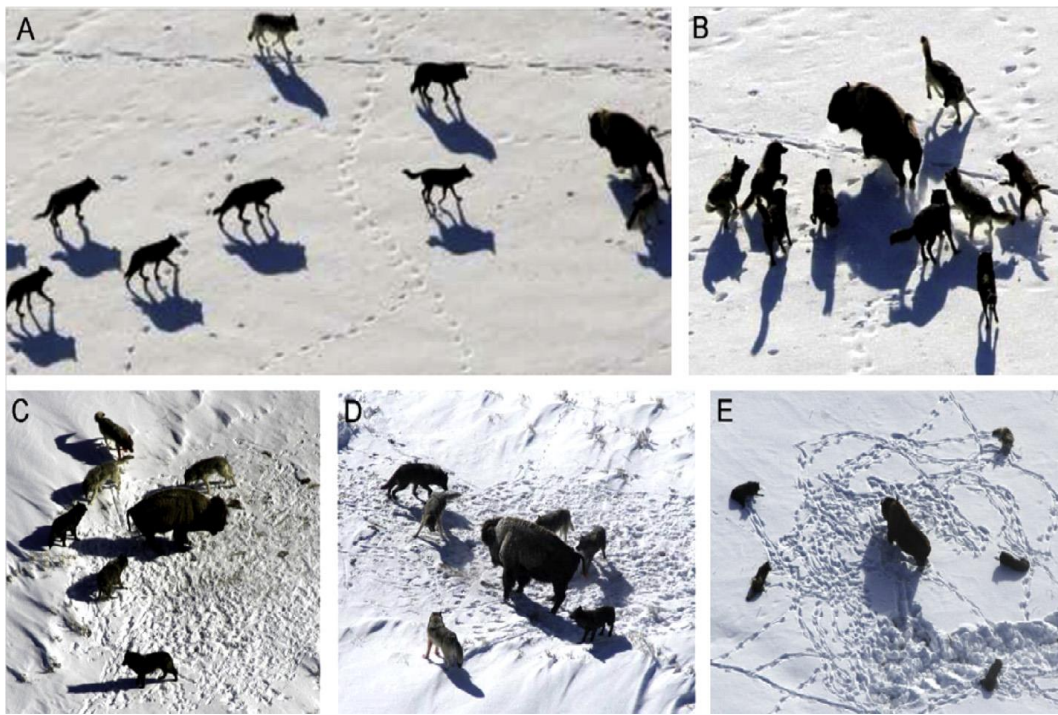
**Figure 3.4.** Grey wolf hierarchy (dominance decreases from alpha to omega) (Mirjalili and et al., 2014).

The leader of the wolves can be a male or a female and it's called alpha. The job of the leader is making judgments on the group in choosing sleeping place, hunting, time to wake that is considered as an order to the group members. The alpha member is also considered to be the dominant member since his/her decisions should be followed by the group (Mech, 1999). The betas which are at the second level in the hierarchy they are assistant wolves that helps the alpha in choosing judgments. The third level of wolves in the hierarchy is called subordinate (or delta). Their jobs are Scouting, sentinels, hunting, caretakers and elders. If the wolves are not alpha, beta, or delta then it's called omega that are at the last level of grey wolf hierarchy, their job is to be scapegoat and sometimes babysitters. Circulation around the victim, hunting and attacking victim, all these three main steps are considered to be the strategy of searching for the victim.

The main strategies of hunting for the grey wolves depending on (Muro and et al., 2011) are shown below:

- Tracking the victim, chasing the victim, and coming near to the victim.
- Keep track of the victim, circulation around the victim, and teasing the victim until the moving stops.
- Attacking the victim.

The steps are shown in figure 3.5.



**Figure 3.5.** Hunting strategy of grey wolves: (A) chasing the victim, coming near to the victim, and tracking the victim (B–D) pursuing the victim, teasing the victim, and circulation around the victim (E) position of attacking (Mirjalili and et al., 2014).

Hunting strategies of grey wolves is modeled mathematically for designing GWO and perform optimization.

- When the hunting process of grey wolves starts the victim will be encircled. The encircling strategy is mathematically modeled in Equation 3.1 and 3.2.

$$D = |C \cdot X_p(t) - A \cdot X(t)| \quad (3.1)$$

$$X(t + 1) = X_p(t) - A \cdot D \quad (3.2)$$

Where  $t$  is indicating the current iteration,  $C$  and  $A$  are indicating to be the coefficient vectors,  $X_p$  indicates the victims' position vector and  $X$  is the grey wolves' position vector. Calculation of the vectors  $C$  and  $A$  are given by the Equation 3.3 and 3.4.

$$A = 2a \cdot r_1 - a \quad (3.3)$$

$$C = 2 \cdot r_2 \quad (3.4)$$

The ingredients of  $a$  are decreased linearly from 2 to 0 during the process of iterations (in both exploitation and exploration phases) and  $r_1, r_2$  are randomly chosen vectors between  $[0, 1]$ .

- The hunting process is generally directed by the alpha member. Occasionally the beta agent and delta agent might take a part in the process. When the hunting strategy of grey wolves mathematically modeled, it could be assumed that the alpha, beta and delta have more information about the possible place of the victim. The first three results are saved and the other members are have to change their positons depending on the best search members' position as shown in the Equation 3.5, 3.6 and 3.7.

$$\begin{aligned} D_\alpha &= |C_1 \cdot X_\alpha - X| \\ D_\beta &= |C_2 \cdot X_\beta - X| \\ D_\delta &= |C_3 \cdot X_\delta - X| \end{aligned} \quad (3.5)$$

$$\begin{aligned} X_1 &= |X_\alpha - A_1(D_\alpha)| \\ X_2 &= |X_\beta - A_2(D_\beta)| \\ X_3 &= |X_\delta - A_3(D_\delta)| \end{aligned} \quad (3.6)$$

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \quad (3.7)$$

The hunting process for grey wolves finishes after the victim stops moving and wolves attack.  $A$  is a vector that can be a randomly chosen value between  $[-2a, 2a]$ , and during the iterations also  $a$  is decreased from 2 to 0.

The operation of exploration in grey wolf optimizer is used according with the position, and that leads the wolves to diverge in between them for searching victim and converge to attacking the victim. The exploration operation is modeled mathematically by utilizing  $A$  with random a bigger value than one or less than -1 to make the search members to diverge from the victim. When  $|A| > 1$ , the wolves will be obliged to diverge from the victim to find a fitter victim.

Steps of GWO for solving TSP is described below.

**Step 1:** Add a standard dataset from TSPLIB.

**Step 2:** Population of the wolves will be initialized.

**Step 3:** Initialize  $a, A$  and  $C$

**Step 4:** Compute fitness of search agents and define the best three first search agents  $X_\alpha, X_\beta$  and  $X_\delta$ .

**Step 5:** For every search agent by using Eq (3.7), update the position of the current search agent.

**Step 6:** Update  $a, A$  and  $C$

**Step 7:** Compute fitness for all search agents and update  $X_\alpha, X_\beta$  and  $X_\delta$

**Step 8:** Apply pair-wise swap mutation

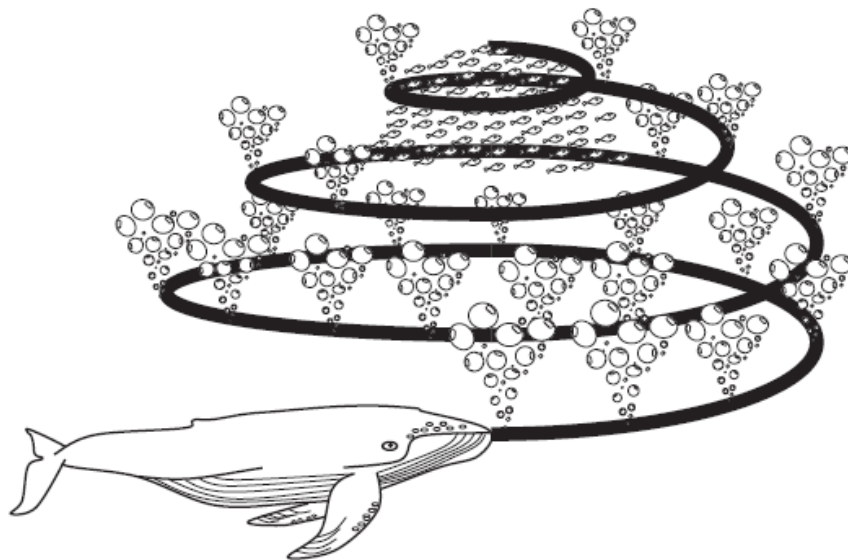
**Step 9:** Optimize population.

**Step 10:** Go back to step 5 until reaching the termination criterion.

**Step 11:** The best solution  $X_\alpha$  will be taken as a final result.

### 3.2.2 Whale Optimization Algorithm (WOA)

The WOA is a novel meta-heuristic algorithm presented by Seyedali Mirjalili in 2016 (Mirjalili and Lewis, 2016). WOA is considered as population based method mimicking the social behavior of humpback whales. This method is taken from the strategy of bubble-net hunting of humpback whales when hunting their victims. Whales are considered to be the biggest mammals in the world they can reach up to thirty meter long and weight more than hundred tons. There are seven various main types of whales existing in the world and humpback is one of them. Whales are mainly considered to be predators. Whales live in groups or alone and they are able to communicate, think, judge, learn, and even they have ability to be emotional like humans but in less cleverness degree. Humpback whales have unique strategy in hunting named as “bubble-net feeding strategy” (Watkins and Schevill, 1979). This foraging behavior is achieved by releasing special bubbles in a spiral shape or (9 shape) as shown in Figure 3.6. Until 2011, they could investigate this behavior only from the surface. However, according to (Goldbogen and et al., 2013) they could investigate this behavior using tag sensors. They discovered two types of maneuvers related with bubble and called them ‘upward-spirals’ and ‘double-loops’. First type of maneuver is humpback whales start diving around 12 m down after that swim up toward the surface with releasing bubble around the victim in a spiral shape. The second maneuver includes the following three various stages: coral loop, lobtail, and capture loop.



**Figure 3.6.** Humpback whale using Bubble-net technique to hunt (Mirjalili and Lewis, 2016).



Hunting strategy of humpback whales is modeled mathematically for designing WOA and performs optimization.

Humpback whales have the ability to locate their victim and encircle them. They consider the current best candidate result is best acquired result and near to the optimal result. When assigning the best search member, the other search members will start updating their positions towards the best search member as described in the Equation 3.9 and 3.10.

$$D = |C \cdot X^*(t) - X(t)| \quad (3.9)$$

$$X(t + 1) = X^*(t) - A \cdot D \quad (3.10)$$

Where  $t$  is indicating the current iteration,  $C$  and  $A$  are considered to be the coefficient vectors,  $X^*$  indicates the position vector of the best result obtained so far,  $X$  indicates the position vector,  $\cdot$  is the multiplication of element-by-element, and  $||$  is the absolute value.

Calculation of the vectors  $C$  and  $A$  are given by the following Equation 3.11 and 3.12.

$$A = 2a \cdot r_1 - a \quad (3.11)$$

$$C = 2 \cdot r_2 \quad (3.12)$$

$A$  value will be decreased linearly from 2 to 0 during the process of iterations (in both exploitation and exploration phases) and  $r_1, r_2$  are randomly chosen vectors between  $[0, 1]$ .

- Bubble-net attacking strategy (exploitation phase)

The humpback whales hunt their victims with the Bubble-net attacking mechanism. This mechanism is mathematically modeled as follow:

Shrinking encircling mechanism; where in this method the value of  $A$  will be chosen randomly in between  $[-a, a]$  and the value of  $a$  will be a decreasing number from 2 to 0 during the process of iterations as shown in Equation 3.11.

Spiral updating position mechanism: in this mechanism firstly the distance will be measured between the whale location and the victim after that the helix-shaped movement of humpback whales is generated as described in the Equation 3.13.

$$X(t + 1) = D'X^*(t).e^{bl}.\cos(2\pi l) + X^*(t) \quad (3.13)$$

Where  $D'$  is the distance between the victim (best solution) and the  $i^{th}$  whale,  $b$  considered as constant used to define the shape of the logarithmic spiral,  $l$  is a randomly chosen number between  $[-1, 1]$  and the  $.$  symbol is the multiplication of element-by-element.

The humpback whales used the mentioned two mechanisms while they swim near to the victim. We set the mathematical model of these two mechanisms; we assume that there is a chance of 50% to select one of the two mechanisms to change the position of whales by the following Equation 3.14.

$$X(t + 1) = \begin{cases} X^*(t) - A.D & \text{if } p < 0.5 \\ (D'X^*(t).e^{bl}.\cos(2\pi l) + X^*(t)) & \text{if } p \geq 0.5 \end{cases} \quad (3.14)$$

The value of  $p$  is a randomly selected number in between  $[0,1]$ .

- Exploration phase. Search for victim

In the exploration stage, the humpback whales (search members) search for victim (best solution) in a random way and update their positions depending to the positions of other whales. In order to oblige the search members to shift far away from reference whale, we take  $A$  with values  $>1$  or  $<1$ .

The exploration phase mathematically modeled by the following Equation 3.15 and 3.16.

$$D = |C.X_{rand} - X| \quad (3.15)$$

$$X(t + 1) = X_{rand} - A.D \quad (3.16)$$

Where  $X_{rand}$  is a randomly chosen position vector (a random whale) taken from the current population.

Steps of WOA for solving TSP is described below.

**Step 1:** Add a standard dataset from TSPLIB.

**Step 2:** Population of the whales will be initialized.

**Step 3:** Compute the fitness of search members and define  $X^*$  as the best search member.

**Step 4:** For every search member update  $a, A, C, l$  and  $p$

**Step 5:** Update the position of the current search member by the Equations (3.9), (3.13) and (3.16) depending on the value of  $a$  and  $p$ .

**Step 6:** Calculate fitness for all search members and update  $X^*$

**Step 7:** Apply pair-wise swap mutation

**Step 8:** Optimize population.

**Step 9:** Go back to step 4 until reaching the termination criteria.

**Step 10:** Choose the best result  $X^*$  as a final result.

### 3.2.3 Particle swarm optimization

The PSO has been developed in 1995 by Eberhart and Kennedy (Eberhart and Kennedy, 1995). PSO mimics the social behaviours of fish schooling and birds flocking as shown in Figure 3.7. The system of PSO starts with a population of random agents. The named “swarm” is called for the population, while, the possible solutions are called as “particles”. The Particles flow in the multidimensional search space for searching of the optimal solution by updating each particles position depending on the experience of the neighbouring particles and its own experience. At the flow time, the current position of the  $i$ th particle is defined by a vector  $X_i = (X_{i1}, X_{i2}, X_{i3}, \dots, X_{iD})$ , where  $D$  indicates the dimensions of the search space. The  $i$ th particles velocity is defined as  $V_i = (V_{i1}, V_{i2}, V_{i3}, \dots, V_{iD})$ . The particles previous best position is saved as the personal best position and named as  $p_{best}$ . The best position acquired by the population so far saved as global best and named as  $g_{best}$ . Optimal solution in PSO is searched by changing the position and the velocity of every particle depending to the Equation 3.17 and 3.18.

$$v_{id}^{t+1} = w * v_{id}^t + c_1 * r_{1i} * (p_{id} - x_{id}^t) + c_2 * r_{2i} * (p_{gd} - x_{id}^t) \quad (3.17)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (3.18)$$

Where  $t$  is the current iteration.  $d \in D$  points to the  $d$ th dimension in the search space.  $w$  is the inertia weight, that is applied to rule the effect of the previous velocities on the current velocity.  $c_1$  and  $c_2$  are acceleration constants.  $r_{1i}$  and  $r_{2i}$  are random vectors in interval  $[0, 1]$ .  $p_{id}$  and  $p_{gd}$  declares the  $p_{best}$  and  $g_{best}$  in the  $d$ th dimension. Steps of PSO are described below.

Step 1: Initialization.

Step 2: Compute the velocity according to Eq (3.17).

Step 3: Update position of particles according to Eq (3.18).

Step 4: Update  $p_i$  if the new  $x_i^{t+1}$  is superior than  $p_i$ .

Step 5: Update  $p_g$  if the new  $x_i^{t+1}$  is superior than  $p_g$ .

Step 6: Go to step 2 until reaching the termination criteria.

Step 7: The global best result  $p_g$  will be taken as a final result.



**Figure 3.7.** An example of PSO (URL13).

The population size number (particles number) in PSO is defined by the user; in step 1 (initialization step) each number of the particles will be given a random velocity and a random solution. When the initial stage starts, a random tour will be assigned to  $p_{id}$ . While for the  $p_{gd}$  (best global solution) will be defined as the best tour depending on the fitness value. At each iteration process, PSO will calculate a new velocity for every particle in the swarm using Eq. (3.17) and then by using Eq (3.18) particles next position will be found as mentioned above in Step three. The new solution  $x_{id}^{t+1}$  the fitness of the particle will be updated for and compared with fitness of  $p_{id}$  and  $p_{gd}$ . For the new solution  $x_{id}^{t+1}$  the fitness value of the particle will be updated and then it will be used to compare it with fitness value of  $p_{id}$  and  $p_{gd}$ . In step four  $P_i$  will be updated with  $x_{id}^{t+1}$  if it is obtained to be better than  $P_i$ . In similar way at step five,  $p_{gd}$  will be updated with  $x_{id}^{t+1}$  if it is obtained to be better than  $p_{gd}$ . In step six (termination criterion) until reaching the last iteration (or obtaining a good fitness of  $p_{gd}$  ) which is the stopping criteria for the optimization problem, step 2, 3, 4 and 5 will be repeated. In particle swarm optimization, best global solution ( $p_{gd}$ ) contains the best solution found by the particles during the operation

process; thus, in step seven the solution of  $p_{gd}$  when reaching the termination criteria will be considered as the outcome of the optimization problem.

In recent years PSO technique developed successfully to be used for discrete and continuous optimization problems to find optimal solutions through local and global models. The method mentioned above is appropriate for problems of continuous value and it can't be used directly to solve problems of discrete value such as TSP. In many studies developers could redefine the basic PSO algorithm by suggesting new concepts one of them is taken from the 'Swap operator' and 'Swap sequence', as in (Wang and et al., 2003).

### 3.2.3.1 Swap operator and swap sequence based operation for TSP

In the Swap Operator (SO) there is a pair of pointers that shows the swap of two cities in a tour. Assume that, having in total five cities in a TSP and there solution is  $R = (a - b - c - d - e)$ . Adding a SO to the R solution would be like

$$R = R + SO(1,3) = (a - b - c - d - e) + SO(1,3) = (c - b - a - d - e).$$

As we can observe here the Swap Operator is more like the mutation process in genetic algorithm which is very strict in solving TSP. The "+" indicates to adding the SO to the solution R.

When applying one or more SO(s) to a solution one after another continually it leads to what called Swap Sequence (SS). In the PSO the SS is considered as the velocity which is described in the following Equation 3.19.

$$SS_{12} = (SO_1, SO_2, SO_3, \dots, SO_n) \quad (3.19)$$

Where  $(SO_1, SO_2, SO_3, \dots, SO_n)$  are indicating to the SO(s). The order of SO(s) in SS is very important when applying to a solution. For example achieving solution  $S_2$  is done by adding  $SS_{12}$  on  $S_1$  which is shown in Equation 3.20.

$$S_2 = S_1 + SS_{12} = S_1 + (SO_1, SO_2, SO_3, \dots, SO_n) \quad (3.20)$$

That order of SO(s) in SS plays an important rule in giving the solutions in a way changing any position of the SO(s) might give a solution that differ from the original solution. Using the Equation 3.21  $SS_{12}$  also can be obtained from the solution  $S_2$  and  $S_1$

$$SS_{12} = S_2 - S_1 = (SO_1, SO_2, SO_3, \dots, SO_n) \quad (3.21)$$

The “-” indicates that SO(s) of  $SS_{12}$  have to be used on solution  $S_1$  to obtain  $S_2$ . As an example, if  $S_1=(a-c-e-b-d)$  and  $S_2=(b-c-a-e-d)$  then  $SS_{12}=SO(1,3), SO(2,3), SO(4,5)$  Moreover, merging two swap operator or more SSs to obtain a new SS. If we suppose that –  $SS_1 = (SO(1,2), SO(5,4), SO(5,1))$  and  $SS_2 = (SO(1,3), SO(5,1), SO(2,1))$  then  $SS_1 \oplus SS_2 = (SO(1,2), SO(5,4), SO(5,1), SO(1,3), SO(5,1), SO(2,1))$

Sometimes when applying more than one SSs and the result is the same solution the SS with least SOs will be considered as basic Basic Swap Sequence (BSS). For example if we have  $SS1=(SO(2,3), SO(3,2), SO(1,2), SO(4,5))$  and  $SS2=(SO(1,2), SO(4,5))$  then  $SS2$  will be considered the BSS because it has lease SO and gives same result with  $SS1$ .

The BSS based PSO algorithm is like the original PSO algorithm considering the initialization face (user defined particles number, each number of the particles will be given a random velocity and a random tour solution and fitness calculation of each tour). Among the tours the best one with the fitness value will be assigned to  $p_{gd}$

Swap sequence PSO is considered as the major PSO based technique in solving TSP where it considers the Swap sequence (SS) as the velocity operator to change the solution (tour) to a new solution by applying all the swap operators in SS. In the mentioned PSO to measure the velocity SS of a particle is calculated on its last best solution tour  $p_{id}$  and the best global solution tour  $p_{gd}$  in the defined population size using the following Equation 3.22 and 3.23.

$$v_{id}^{t+1} = v_{id}^t \alpha (p_{id} - x_{id}^t) \beta (p_{gd} - x_{id}^t) \quad \alpha, \beta [1,0] \quad (3.22)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (3.23)$$

Steps of Swap sequence PSO are described below.

**Step 1:** All the components will be initialized.

**Step 2:** In the PSO swarm for every particle  $x_{id}^t$ .

- a) Compute velocity  $v_{id}^t$  depending on equation (3.22)
- b) Update solution using equation (3.23)
- c) Update  $p_{id}$  if the new result  $x_{id}^{t+1}$  is superior than  $p_{id}$
- d) Update  $p_{gd}$  if the new result  $x_{id}^{t+1}$  is superior than  $p_{gd}$

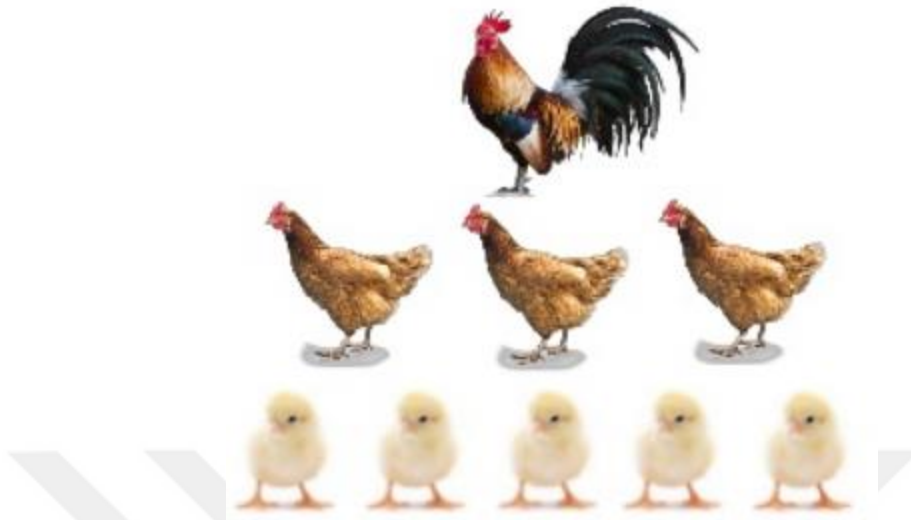
**Step 3:** Go to step 2 until reaching the termination criteria.

**Step 4:** Choose the global best result  $p_g$  as a final result.



### 3.2.4 Chicken swarm optimization

The CSO is a bio-inspired method presented by Xianbing Meng, Yu Liu<sup>2</sup>, Xiaozhi Gao, and Hengzhen Zhang in 2014 (Meng and et al., 2014). Home Chickens are considered at the top list of the most widespread birds in the world, human use chicken eggs as main source of food. They mimic the social hierarchal order in the chicken swarm and the chicken swarm behavior. Home Chickens are like most of the birds they live together in flocks. Chickens can realize more than hundred individuals even after many months of separation which refer to their sophisticated cognition. To communicate between them chickens use more than 30 distinct sounds, which range from cries, chirps, cackles and clucks, which contains many information associated to food discovery, nesting, danger and mating. Chickens can learn from their trials and errors, when deciding something also they have ability to take information from their previous experiences and others' [URL6]. The chicken swarm can be splitted into many groups, which every group builds on one rooster and many hens and chicks as shown in Figure 3.8. The hierarchal order of the swarm plays an important function in the social lives of those birds. The predominant chickens in a group will dominate the weak individuals. The existence of dominant chickens means extra dominant hens will stay around the head roosters. Adding or removing chickens from existed groups will lead to a temporary disarrangement in the social arrangement of the swarm until a particular hierarchal order is established again [URL7]. The priority to access food is done by the dominant individuals and sometimes when roosters find food they might invite their group-mates to be the first eaters. The same behavior also hens apply it when they raise their children. When other chickens from other group exceed the territory of group the roosters will emit a loud call [35]. In general, the behavior of the chickens varies with gender. The head rooster's main job is searching for food, and fighting with other chickens that exceed their territory. While the dominant chickens will stay near to the head roosters to forage for food and the chicks job is searching for the food around their mother. The submissive individuals would unwillingly stay around the group to search for food.



**Figure 3.8.** Hierarchical order of chicken swarms (dominance decreases from rooster to chicks) (URL14).

CSO can be developed mathematically according to the chickens' behaviors by taking in consider the rules below.

1. In the chicken swarm, there are many groups and every group consists of dominant rooster, a number of hens, and some chicks.
2. Dividing the chicken swarm into many groups and determining the identity of the chickens (roosters, hens and chicks) all depends on the chicken's fitness values. The best fitness values of the chickens would be assigned as roosters and each one will be a head rooster in a group. For the chicks the worst fitness values of the chickens will be assigned to them. The rest of the chickens would be the hens. The hens decide choosing to live in a group randomly. The relationship of mother-child (hens and chicks) also will be established randomly.
3. The order of hierarchical, mother-child relationship and dominance relationship will stay unchanged in a group. It can be updated only every many ( $G$ ) time steps.
4. Rooster will be followed in the group by chickens to look for food; also they might block other individuals from stealing their own food. The chicks stay around their mothers (hens) to search for food. Competition for food would be advantaged to the dominant individuals.

Assuming that the number of the roosters, the hens, the chicks and the mother hens would be shortened as RN, HN, CN and MN respectively. The best RN chickens are the roosters, and the worst CN chickens are the chicks. Minimal fitness values of RN correspond to the best RN chickens.

### 3.2.4.1 The process of chicken's movement

Roosters that own best fitness values would have the priority to access the food more than the other roosters that have bad fitness values. More briefly, roosters that have better fitness values have the ability to search for food more than other roosters with worse fitness values in a wide range areas. This can be formulated by the following Equation 3.24 and 3.25.

$$x_{i,j}^{t+1} = x_{i,j}^t * (1 + randn(0, \sigma^2)) \quad (3.24)$$

$$\sigma^2 \begin{cases} 1, & \text{if } f_i \leq f_k, \\ \exp\left(\frac{f_k - f_i}{|f_i| + \varepsilon}\right), & \text{otherwise, } k \in [1, N], k \neq i \end{cases} \quad (3.25)$$

Where  $randn(0, \sigma^2)$  is a Gaussian distribution with mean equal to 0 and standard deviation  $\sigma^2$ .  $\varepsilon$ , indicates to a constant which is the smallest value in the computer that's used to prevent zero-division-error,  $k$ , is the index of the rooster's, which is selected in a random way from the group of the roosters,  $f$ , is the fitness value of the corresponding  $x$ .

And about the hens, they could track their mates in the group (roosters) for searching food. Furthermore, they have the ability to take the best food randomly found by other individual chickens, although they would be repressed for stealing the founded food by the other chickens. Having many dominant hens in the group will lead to advantage in competition for finding food which is better than submissive ones. Those behaviors could be mathematically modeled by the following Equations 3.26, 3.27 and 3.28.

$$x_{i,j}^{t+1} = x_{i,j}^t + S_1 * rand(x_{r_1,j}^t - x_{i,j}^t) + S_2 * rand(x_{r_2,j}^t - x_{i,j}^t) \quad (3.26)$$

$$S_1 = \exp((f_i - f_{r_1}) / (abs(f_i) + \varepsilon)) \quad (3.27)$$

$$S_2 = \exp((f_{r_2} - f_i)) \quad (3.28)$$

Where *rand* is a randomly chosen number between [0, 1],  $r_1 \in [1, \dots, N]$  is an index of the rooster, that is the *i*th hen's group-mate,  $r_2 \in [1, \dots, N]$  is the index of the individual chicken (rooster or hen), that is selected in a random way from the chicken swarm.  $r_1 \neq r_2$ .

Obviously,  $f_i > f_{r_1}, f_i > f_{r_2}$ , so  $S_2 < 1 < S_1$ . Assume that  $S_1 = 0$ , then the *i*th hen try to search for the food and tracked by other individual chickens. The big value of  $S_1$  means the variation in the fitness values of the two chickens, the small value of  $S_2$  means the gap between the positions of the two chickens are bigger. So the hen's individuals cannot take the found food by the other chickens in easy way. The purpose of why the formulation of  $S_1$  and  $S_2$  is different from each other is that the individuals in a group will be in competition always. In a simple way, the chickens' fitness values that is proportional to the rooster' fitness value are simulated in a group as the competitions between chickens. Assume  $S_2 = 0$ , then the *i*th hen will try look for food in the area that they belong. For a particular group, the fitness value of the rooster is exclusive. So the smaller fitness value of the *i*th hen, the nearer  $S_1$  approximates to 1 and the smaller gap in the positions between the *i*th hen and its group-mate rooster is. Having a lot of dominant hens is better than having submissive hens to eat the food. The chicks change their places to be near to their mother to search for food. This phenomenon can be mathematically formulated by the following Equation 3.29.

$$x_{i,j}^{t+1} = x_{i,j}^t + FL * (x_{m,j}^t - x_{i,j}^t) \quad (3.29)$$

Where  $x_{m,j}^t$  is the *i*th chick's mother position ( $m \in [1, N]$ ). ( $FL \in [0, 2]$ ) is a parameter, which indicates the chick would track its mother to search for food. Considering the individual differences, the *FL* of each chick is chosen in a random way between [0, 2].

Steps of CSO for solving TSP is described below.

**Step 1:** Add a standard dataset from TSPLIB.

**Step 2:** Population of chickens will be initialized and the related parameters define will defined.

**Step 3:** Calculate fitness of the chickens.

**Step 4:** If its new generation then fitness values of the chickens' will be ranked and a hierarchal order in the swarm will be established; the swarm will be divided into many groups, and the relationship in a group between the chicks and mother hens will be checked.

**Step 5:** Depending on the value of  $i$ .

- a) Update roosters solution/location according to Equation (3.24)
- b) Update hens solution/location according to Equation (3.26)
- c) Update chicks solution/location according to Equation (3.29)

**Step 6:** Calculate the new obtained result; If the new obtained result is superior than its previous one, update it;

**Step 7:** Apply pair-wise swap mutation

**Step 8:** Optimize population.

**Step 9:** Go back to step 4 until reaching termination criteria.

**Step 10:** The best result rooster will be taken as an outcome.

### 3.2.5 Crow search algorithm

CSA is a meta-heuristic algorithm taken from the intelligent behavior of crow birds which is presented by Alireza Askarzadeh in 2016 (Askarzadeh, 2016). Crows are genus of birds that are widely distributed around the world and now they belong to the list of world's most intelligent animals [URL9, URL10, URL11]. According to their body size (brain-to-body ratio) they contain the largest brain which is different than a human brain in a slight low way. Most of the crows live in groups; and they show very gorgeous examples of intelligence and usually they record very high results on intelligence tests. They can communicate in sophisticated ways, use tools, memorize faces and store and restore food between seasons (URL9, URL10, URL11) (Prior and et al., 2008) In the flock crows perform a behavior that's similar to optimization process. That behavior shows that these birds store their excessed food in specific positions (hiding places) in the area that they live and when they want that stored food they take it back. Crows are considered to be greedy birds when they try to find better sources of food they follow and watch each other and other birds to observe the location of the hidden food by other birds, and steal that hidden food at the moment when the owner departs from that location. When a member of crows commits a thievery, that crow would take in consider more precautions like changing hiding positions to not be a stolen in the future. In fact, crows have the ability to take advantage of their own experience from being a pilferer to predict a pilferers behavior in the future, and to protect their caches to not be pilfered in the future they can determine the safest methods for that (URL10). As we said when crow tries to find a good source of food hidden by a another crow it will follow that crow but it will not be an easy work specially when the crow knows that another crow is tracking him, the crow will try to trick that greedy crow by heading to different position in the area that they live. From the view point of optimization, the member of the crows would be considered as searchers, the living area (environment) is considered as search space, in the living area all the positions are considered to be a feasible solution, the source food quality is considered as objective (fitness) function and the global solution of the problem is the best food source in the environment. Depending on these similarities of the intelligent behavior of the crows, CSA tries to simulate that behavior to find optimization problems solution. Figure 3.9 shows how crows follow each other and steal the hidden food from each other.



Figure 3.9. Stealing strategy of crows (URL15).

The principles of CSA are described as follows:

- Crows live in flocks.
- Crows have the ability to memorize the hiding places positions.
- Crows do thievery by following each other.
- Crows use probability to not being pilfered and protect their caches.

Let's assume that there are number of crows in  $d$ -dimensional environment. The crows number (flock size) will be  $N$  and the crow  $i$  position in the search space at time (iteration)  $iter$  is clarified by a vector  $x^{i,iter}$  ( $i = 1, 2, \dots, N; iter = 1, 2, \dots, iter_{max}$ ) where  $x^{i,iter} = [x_1^{i,iter}, x_2^{i,iter}, \dots, x_d^{i,iter}]$  and  $iter_{max}$  indicates to the maximum iteration number. In the memory of each crow the hiding place position is memorized. For a specific iteration  $iter$ , the hiding place position of crow  $i$  is shown as  $m^{i,iter}$ . This will be considered as the best position obtained by crow  $i$ . so each crow will memorize in his memory the position of his own best experience. Crows try to look for better sources of food (hiding places) by moving in the environment.

Assuming that crow  $j$  at iteration  $iter$ , desires to visit its hiding place,  $m^{j,iter}$ . At that iteration, when crow  $j$  tries to approach to its hiding place crow  $i$  will decide to follow crow  $j$ . At that moment, two situations can happen:

**Situation 1:** Crow  $j$  have no idea that its being followed by crow  $i$ . That will lead crow  $i$  to be near to the hiding place of crow  $j$ . At that situation, the new obtained position of crow  $i$  would be calculated by the following Equation 3.30

$$x^{i,iter+1} = x^{i,iter} + r_i * fl^{i,iter} * (m^{j,iter} - x^{i,iter}) \quad (3.30)$$

Where  $r_i$  is a randomly chosen number between [0,1] and  $fl^{i,iter}$  indicates the flight length of crow  $i$  at iteration  $iter$ .

The situation schematic and  $fl$  effect on the search capability is shown in Figure 3.10. Local search (at the neighborhood of  $x^{i,iter}$ ) comes from small values of  $fl$  and global search (far from  $x^{i,iter}$ ) comes from large values of  $fl$ . As shown in Figure 3.10(a), if the selected value of  $fl$  is smaller than one then the next position of crow  $i$  will be on the dashed line between  $x^{i,iter}$  and  $m^{j,iter}$ . And as Figure 3.10(b) shows, if the selected value of  $fl$  is bigger than one then the next position of crow  $i$  will be on the dashed line that might pass  $m^{j,iter}$ .

**Situation 2:** Crow  $j$  have idea its being followed by crow  $i$ . That will lead crow  $j$  to protect its cache to not be pilfered, and it will try to fool crow  $i$  in a way that it may leave to different position in the search space.

Both Situation 1 and 2 can be described by the following Equation 3.31:

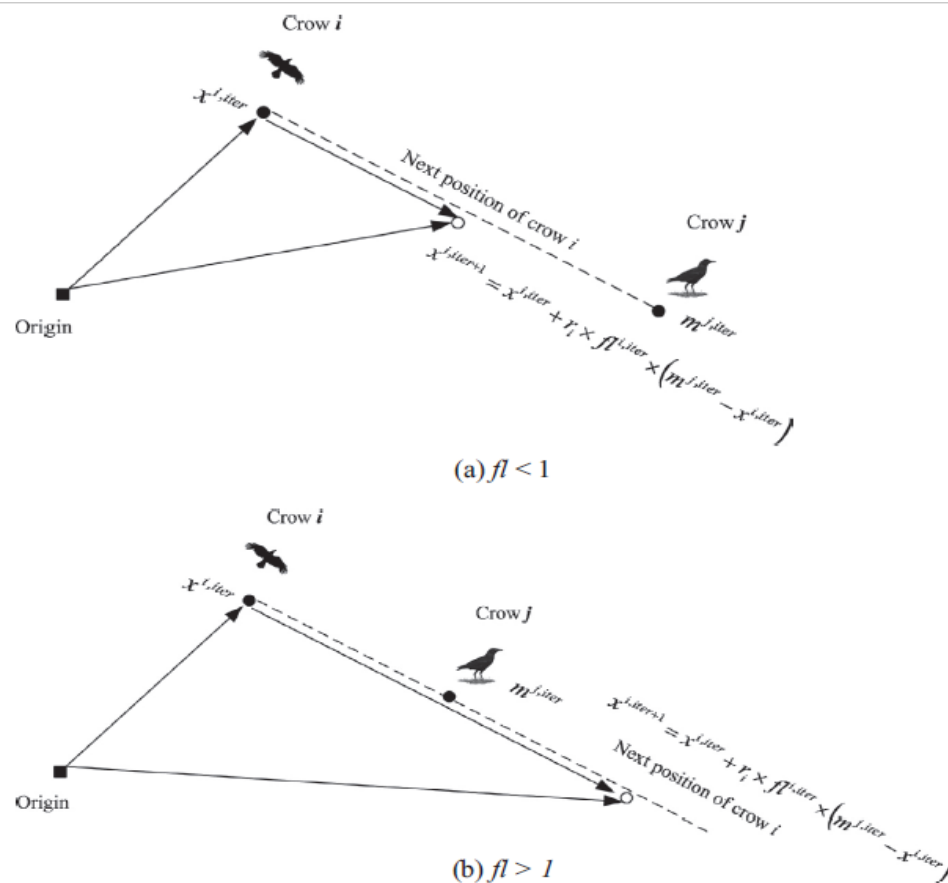
$$x^{i,iter+1} = \begin{cases} x^{i,iter} + r_i * fl^{i,iter} * (m^{j,iter} - x^{i,iter}) & r_j \geq AP^{j,iter} \\ a \text{ random position} & \text{otherwise} \end{cases} \quad (31)$$

The  $r_j$  is a randomly chosen number uniformly distributed between [0,1] and  $AP^{j,iter}$  corresponds to the awareness probability of crow  $j$  at iteration  $iter$ .

According to (Yang, 2011) in meta-heuristic algorithms good balance between intensification and diversification should provide. In CSA, parameter of awareness probability (AP) is the basic controller of the intensification and diversification. By decreasing the value of awareness probability, CSA will head to the region of local search where in that region there will be a current good solution. Consequently, when the awareness probability values are small that will lead to increasing the intensification. Furthermore, when the awareness probability values are increased, there will be decreasing in the probability of searching the neighborhood of current good solutions and CSA will head to search the environment on a global scale



(randomization). Therefore, when the awareness probability values are large that will lead to increasing the diversification.



**Figure 3.10.** Flow digram of situation 1 in CSA (a)  $fl < 1$  and (b)  $fl > 1$ . Crow  $i$  have the ability to be in any position on the dashed line (Askarzadeh, 2016).

### 3.2.5.1 Implementation of CSA for optimization is described in the following steps

**Step 1:** Initialize the optimization problem and adjust parameters

First the optimization problem are defined, constraints and decision variables and are defined. Then, parameters of CSA are set (flock size ( $N$ ), flight length ( $fl$ ), awareness probability ( $AP$ ) and maximum number of iterations ( $iter_{max}$ )).

**Step 2:** Initializing crow's position and memory

Crows with size  $N$  are positioned in a random way in a  $d$ -dimensional search space. Each number of crows in the problem indicates a feasible solution and  $d$  is the number of decision variables. Using Equation 3.32 we can initialize crow's position.

$$Crows = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & \dots & x_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^N & x_2^N & \dots & x_d^N \end{bmatrix} \quad (3.32)$$

Each crows' memory is initialized. When the crows at initial iteration the members would have no experiences at all, so it is supposed that their initial positions will be their first position for hiding foods. Using Equation 3.33 we can initialize crow's memory.

$$Memory = \begin{bmatrix} m_1^1 & m_2^1 & \dots & m_d^1 \\ m_1^2 & m_2^2 & \dots & m_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ m_1^N & m_2^N & \dots & m_d^N \end{bmatrix} \quad (3.33)$$

### Step 3: Evaluation of objective (fitness) function

The quality of the positions for every crow is computed by sending the values decision variable into the fitness function.

### Step 4: Generation of the new position

The new positions of the crows in the search space are generated as described in next section: assume crow  $i$  decides to make a new position. For that purpose, crow  $i$  chooses randomly from the flock one of the crows (for example crow  $j$ ) and tracks that crow to find out the foods position hidden by that crow ( $m^j$ ). The new generated position of crow  $i$  is acquired by Eq. (3.31). For the other crows in the flock the process is repeated for each one

### Step 5: Feasibility of the new positions are checked

For each crow in the flock the feasibility of the new positions will be checked. If the generated new position of any crow is better, then crow changes its position. Otherwise, the new generated position will not be considered if it's not the feasible and the crow will stay in its current position.

**Step 6:** Evaluation of objective (fitness) function for the new positions. For each crow the value of fitness function of the new positions are calculated.

**Step 7:** Updating the crow's memory

This process is done by the Equation 3.34:

$$m^{i,iter+1} = \begin{cases} x^{i,iter+1} & f(x^{i,iter+1}) \text{ is better than } f(m^{i,iter}) \\ m^{i,iter} & \text{otherwise} \end{cases} \quad (3.34)$$

Where  $f(.)$  indicates the value of fitness function. For a crow if the obtained new positions fitness function value is superior than its memorized positions' fitness function, then by using that new position it will update its memory.

**Step 8:** Termination criteria is checked

Until reaching  $iter_{max}$  step four to step seven are repeated. After reaching termination criteria, the result of the optimization problem will be the best position of the memory in terms of the fitness function value.

Steps of CSA for solving TSP is described below.

**Step 1:** Add a standard dataset from TSPLIB.

**Step 2:** Initialize the population of crows N and define the related parameters.

**Step 3:** Initialize the memory of each crow.

**Step 4:** In random way choose one of the crows to track another crow and define the awareness probability

**Step 5:** depending on the awareness probability value, change the position of the crow using equation (3.30).

**Step 6:** Loop to step 4 until reaching max N

**Step 7:** Feasibility of new positions will be checked and the new position of the crows will be evaluated

**Step 8:** Update the memory of crows.

**Step 9:** Apply pair-wise swap mutation

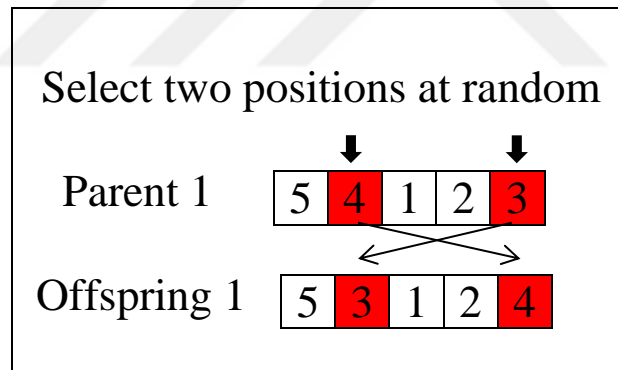
**Step 10:** Optimize population.

**Step 11:** Go back to step 4 until reaching termination criteria.

**Step 12:** Choose the best solution mem as a final result.

### 3.2 Pairwise Swap Mutation

In order to improve our algorithms Pairwise swap mutation (PSM) method were added to the algorithms to increase their performance. PSM method was proposed by Banzhaf in 1990. The mechanism of this method is to select two cities at a random way from the obtained tour and swap those two cities (Banzhaf, 1990). This method also has different names like random swap exchange mutation and interchange mutation. In this study Solving TSP by GWO, WOA, CSA and CSO is done by adding PSM for improving the whole position of population as shown in Figure 3.11.



**Figure 3.11.** pair-wise swap mutation (PSM)

#### 4. EXPERIMENTAL RESULTS AND DISCUSSION

All five algorithms are implemented to TSP. The testing platform for the TSP as: Processor type: CPU Intel Core™2 Duo, Frequency: 2.00 GHz, RAM: 4GB, Operating system: Windows 7 32-bit. All five meta-heuristic algorithms are implemented with some parameters modifications in order to be adapted to solve TSP. For CSA the awareness probability of crows AP set to 0.65, the flight length fl set to 1 and for CSO the rooster's percent set to 0.15, hens percent set to 0.7 and chicks percent set to 0.5, and for the other algorithms the parameters were selected randomly as mentioned above in optimization section. Population size in the algorithms set up to 200 to be compared with other scientific literatures and obtain better results, Maximum number of iterations set to 2000 iterations. In this thesis, 6 benchmark problems are used from TSPLIB (URL12) where number of cities varied from 30 to 100.

Table 4.1 summarizes the experiments results and they are averaged with 30 runs of all the models for each data set. The first column of the table shows the name of the dataset with the optimal solution length. In the second column the 'Best value' shows the best solution length achieved after 30 runs, the 'Worst value' shows the worst solution lengths found after 30 runs, Err is the percentage of error, 'time' represents the average time consumed by algorithms. The percentage error of a solution is given by the Equation (4.1):

$$\text{error} = \frac{\frac{\text{Best value} + \text{Worst value}}{2} - \text{Opt}}{\text{Opt}} \quad (4.1)$$

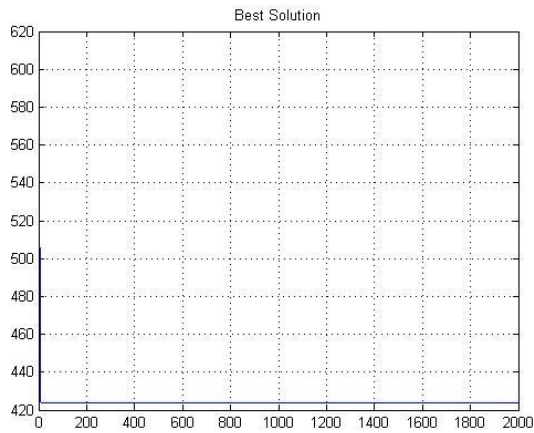
Table 4.1 shows the results obtained by GWO, WOA, CSA, CSO and PSO. It can be observed from the Table 4.1 that the results of dataset with size 30 cities obtained by GWO, WOA, CSA and CSO is the optimal result while PSO was the worst and the obtained solutions of datasets with the size of greater than 30 cities are close to the optimal results and the error percentages of the results are smaller. For Berlin52, Eil51, Eil76, St70, and KroA100 the error percentages are less than 1 for GWO, WOA, CSA and CSO which indicate that GWO, WOA, CSA and CSO can obtain better results than PSO. Among all the algorithms GWO and WOA were more stable and obtained better results than others.

In order to simplify observation, the curve evolution diagram for Oliver30 and Berlin52 with GWO, WOA, CSA and CSO are given in Figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 and 4.8. It can be observed from the figures that GWO and WOA can achieve convergence in very short iterations in Oliver30 while CSA and CSO required more iterations to reach the solution, the iteration number for convergence is less than 50 in GWO and WOA while for CSA and CSO it was more than 50 iterations, and the iteration number for convergence of Berlin52 was less than 400 iterations in all the algorithms with different solutions. Among Figure 4.9 and 4.32 the best obtained tour by GWO, WOA, CSA and CSO of Oliver30, eli51, Berlin52, St70, eli76 and Kro100 are shown.

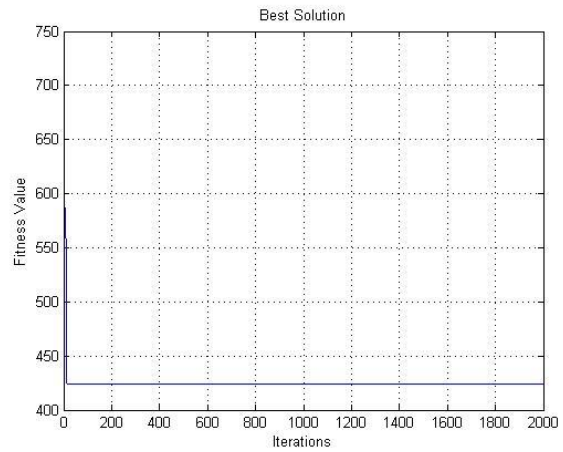
**Table 4.1.** Computational results of GWO, WOA, CSO, CSA and PSO for 6 TSP benchmark datasets of TSPLIB

Dataset	Calculation index	GWO	WOA	CSA	CSO	PSO
Oliver30(423)	Best value	423	423	423	423	801
	Worst value	423	423	457	470	927
	Average	423	423	436	441	862
	Err(%)	0	0	0.04	0.05	1.9
	Time (sec)	1699	1141	580	25145	9
Eil51(429)	Best value	429	437	439	444	907
	Worst value	454	464	482	594	1374
	Average	442	448	451	483	1235
	Err(%)	0.02	0.05	0.07	0.2	2.7
	Time (sec)	3035	2731	966	42240	18
Berlin52(7542)	Best value	7680	7661	7742	8092	16144
	Worst value	8505	8323	8351	8772	23068
	Average	8030	7940	8032	8608	22206
	Err(%)	0.07	0.05	0.06	0.19	1.6
	Time (sec)	3051	2853	1037	43896	19
St70(675)	Best value	684	679	697	776	2001
	Worst value	736	739	767	864	3496
	Average	718	713	731	802	2790
	Err(%)	0.05	0.05	0.08	0.21	4.5
	Time (sec)	4705	4746	1384	81758	40
Eil76(538)	Best value	569	569	571	582	1662
	Worst value	602	614	596	653	2179
	Average	575	587	584	612	2008
	Err(%)	0.088	0.099	0.08	0.14	2.5
	Time (sec)	5446	5257	1572	90546	42
KroA100( 21282)	Best value	21984	21958	22451	24020	114001
	Worst value	25475	25776	24907	27523	191895
	Average	23215	23334	23642	26394	134460
	Err(%)	0.1	0.1	0.11	0.21	6.1
	Time (sec)	8660	9305	2214	140931	108

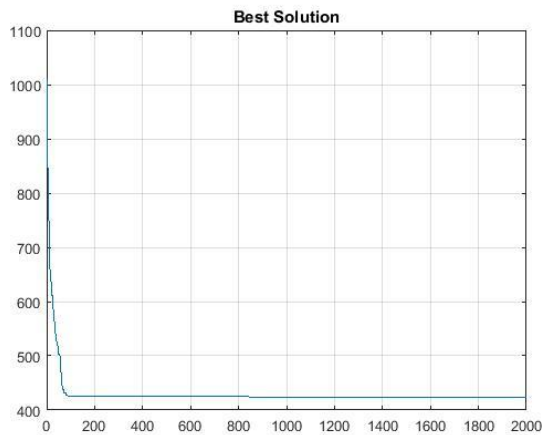
For the working time we can observe that CSA was faster than GWO, WOA and CSO. CSO required a big amount of time to solve TSP even with small datasets due to its complexity.



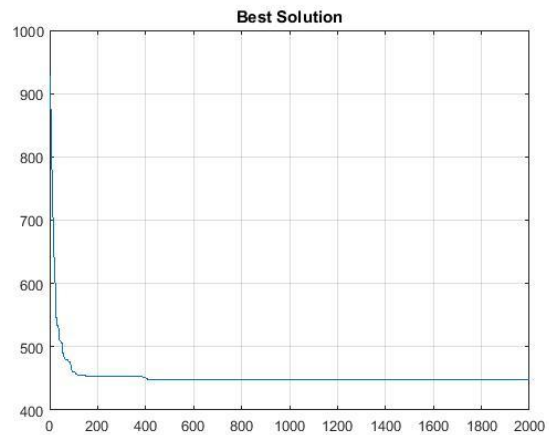
**Figure 4.1.** Plot of best value of Oliver30 by WOA.



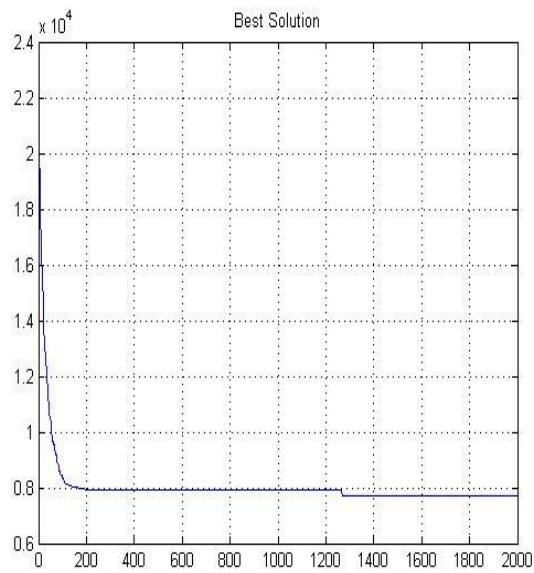
**Figure 4.2.** Plot of best value of Oliver30 by GWO.



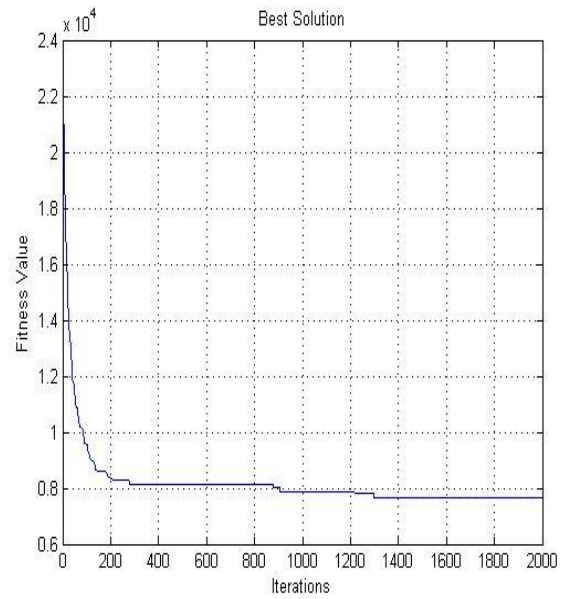
**Figure 4.3.** Plot of best value of Oliver30 by CSA.



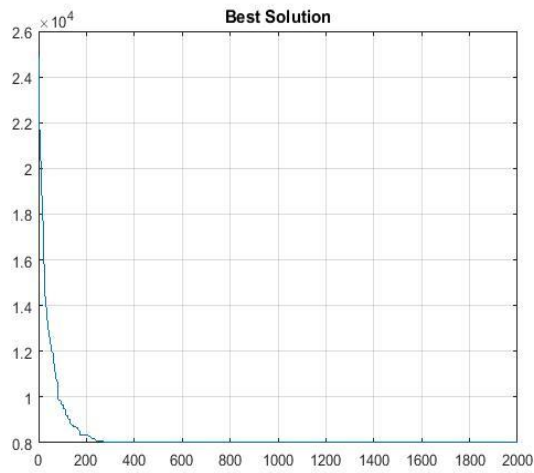
**Figure 4.4.** Plot of best value of Oliver30 by CSO



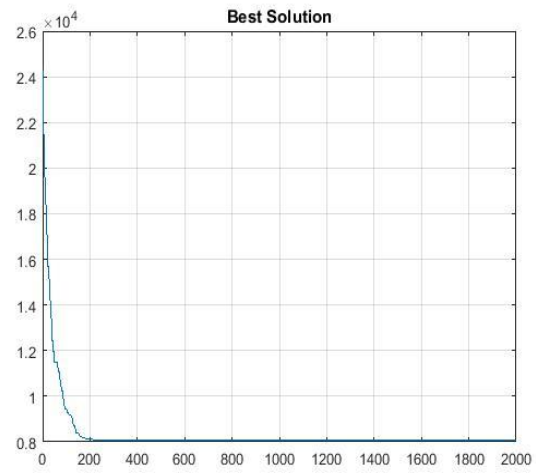
**Figure 4.5.** Plot of best value of Berlin52 by WOA.



**Figure 4.6.** Plot of best value of Berlin52 by GWO.



**Figure 4.7.** Plot of best value of Berlin52 by CSA.



**Figure 4.8.** Plot of best value of Berlin52 by CSO.



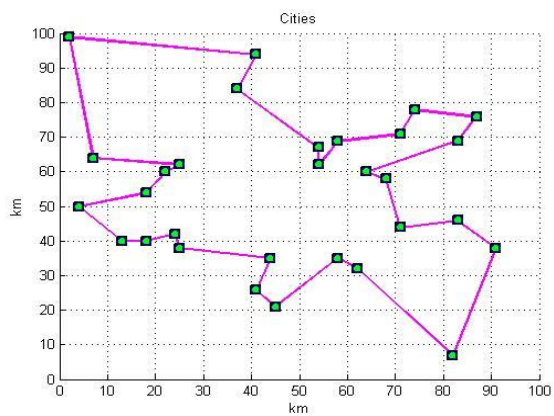


Figure 4.9. Best solution tour of Oliver30 by WOA

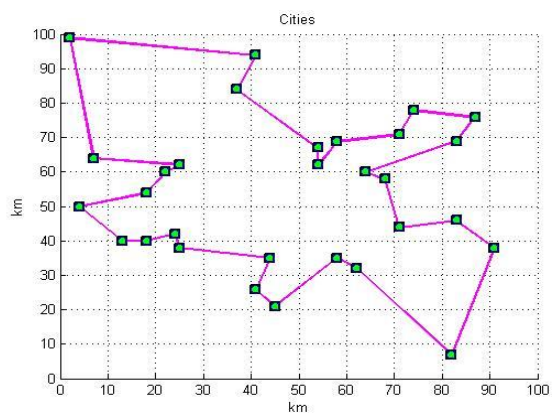


Figure 4.10. Best solution tour of Oliver30 by GWO

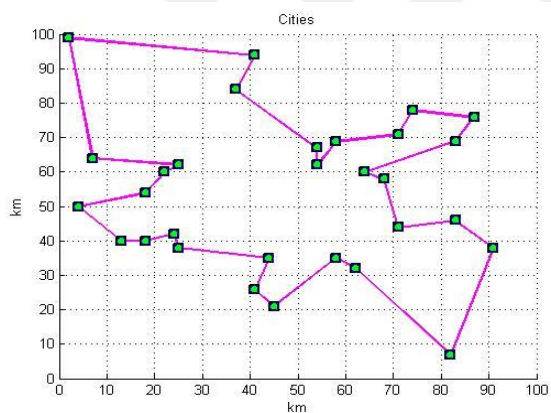


Figure 4.11. Best solution tour of Oliver30 by CSO

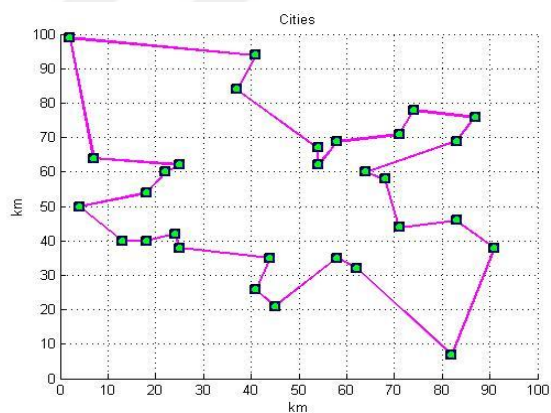


Figure 4.12. Best solution tour of Oliver30 by CSA

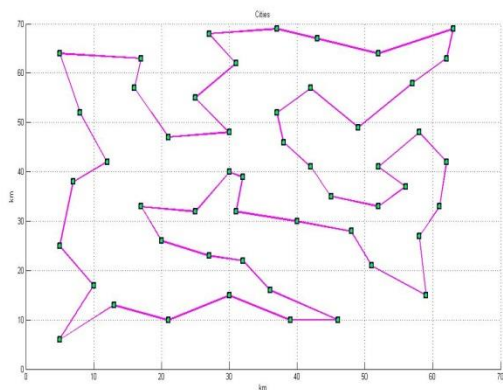


Figure 4.13. Best solution tour of eli51 by WOA

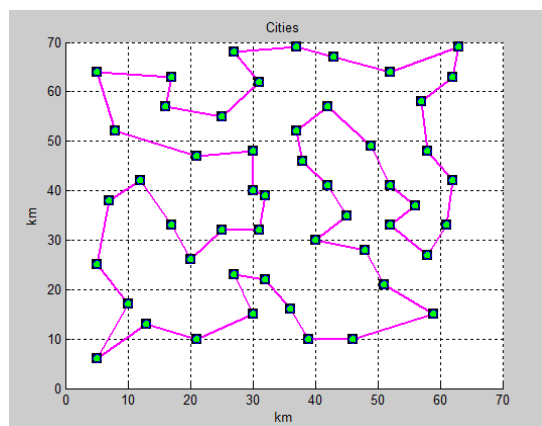


Figure 4.14. Best solution tour of eli51 by GWO

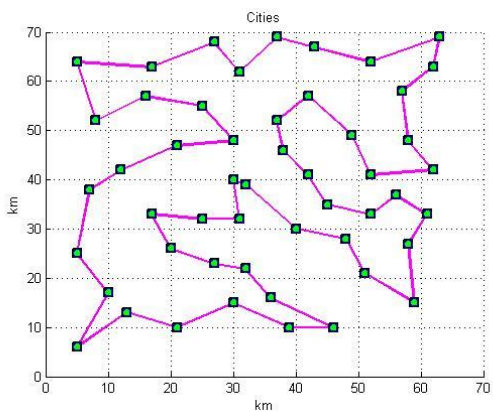


Figure 4.15. Best solution tour of eli51 by CSA

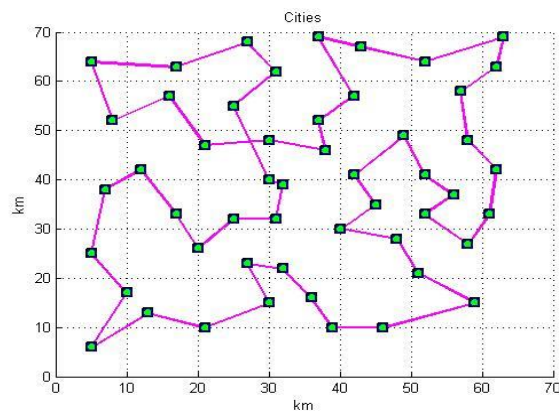


Figure 4.16. Best solution tour of eli51 by CSO

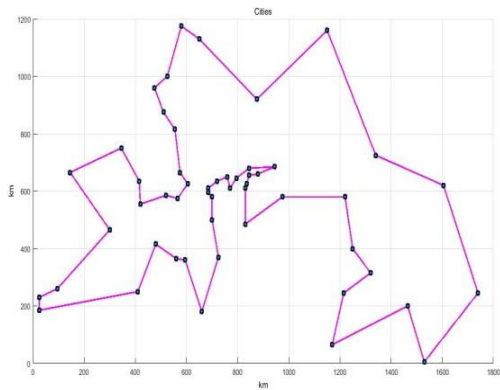


Figure 4.17. Best solution tour of Berlin52 by WOA.

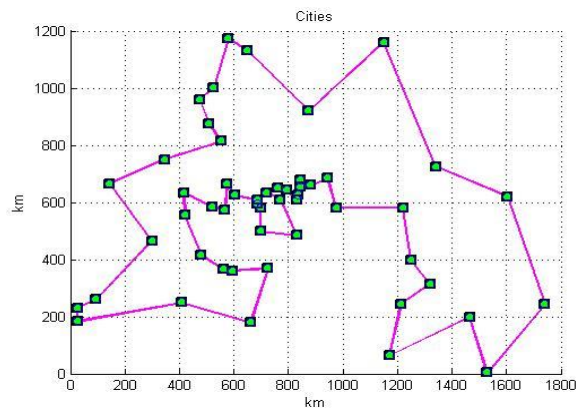


Figure 4.18. Best solution tour of Berlin52 by GOW.

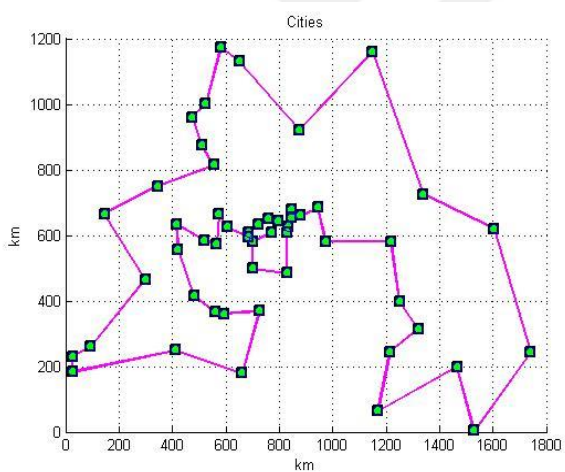


Figure 4.19. Best solution tour of Berlin52 by CSA.

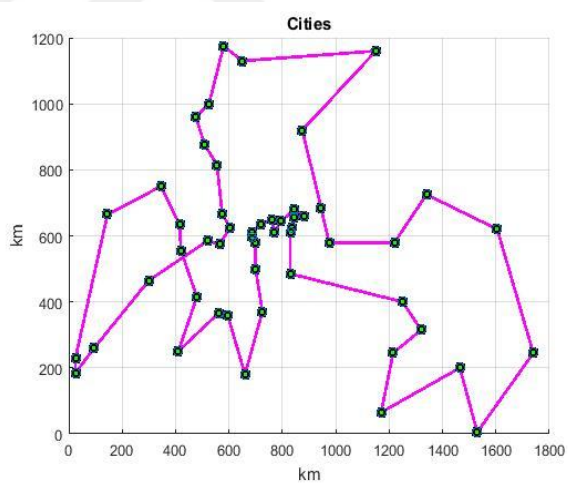


Figure 4.20. Best solution tour of Berlin52 by CSO.

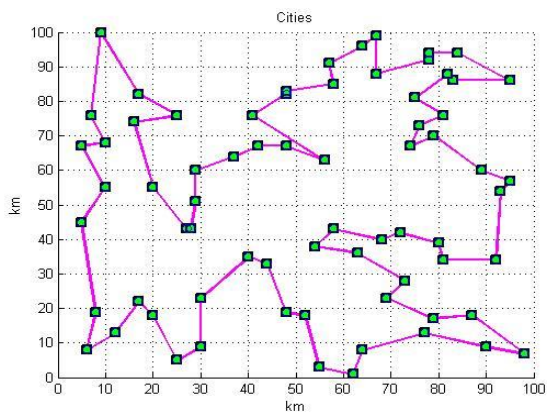


Figure 4.21. Best solution tour of St70 by WOA.



Figure 4.22. Best solution tour of St70 by GOW.

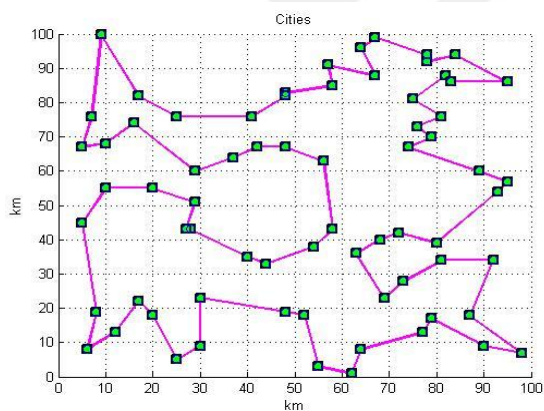
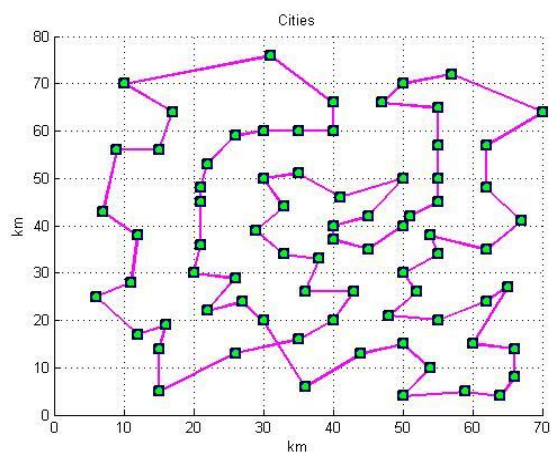


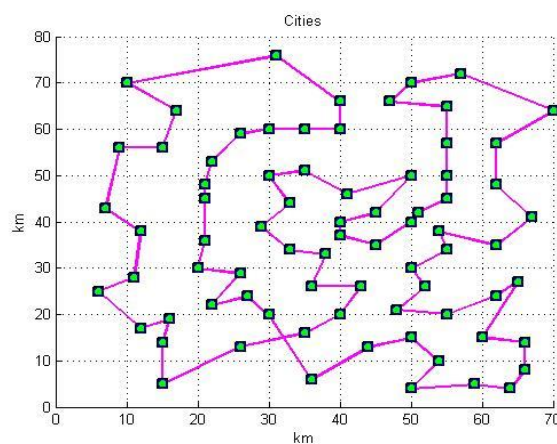
Figure 4.23. Best solution tour of St70 by CSA.



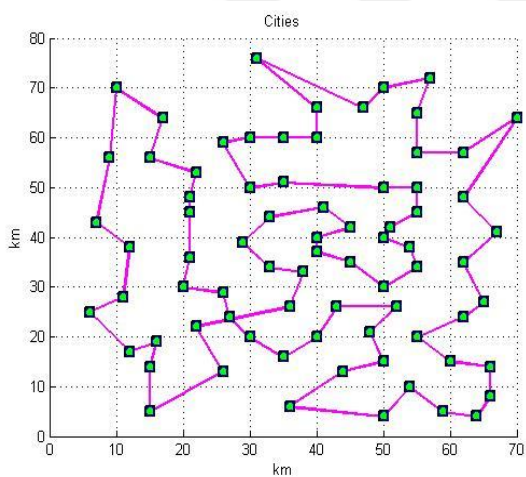
Figure 4.24. Best solution tour of St70 by CSO.



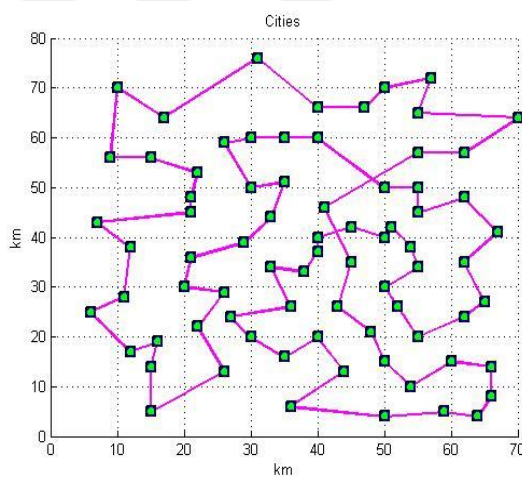
**Figure 4.25.** Best solution tour of eli76 by WOA.



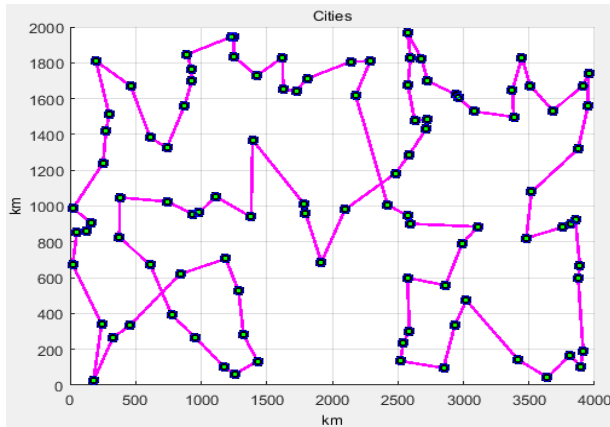
**Figure 4.26.** Best solution tour of eli76 by GOW.



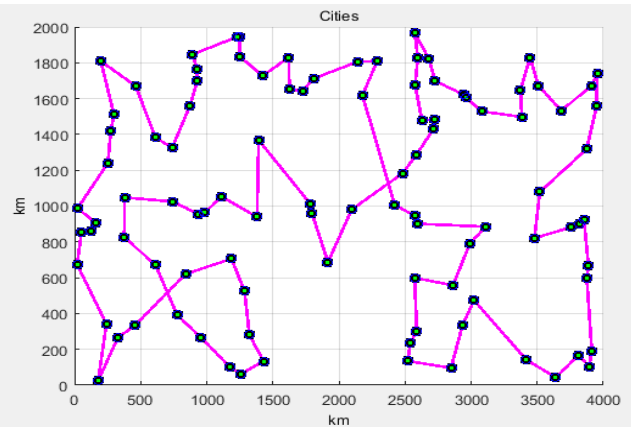
**Figure 4.27.** Best solution tour of eli76 by CSA.



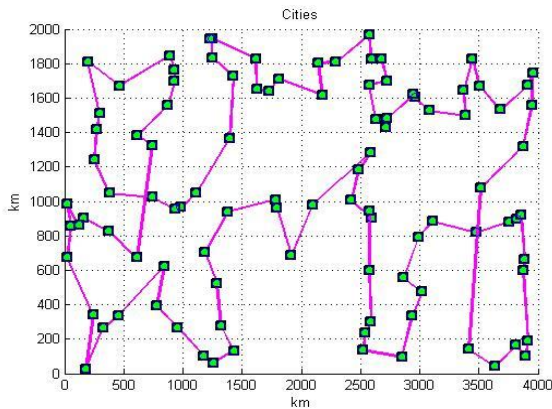
**Figure 4.28.** Best solution tour of eli76 by CSO.



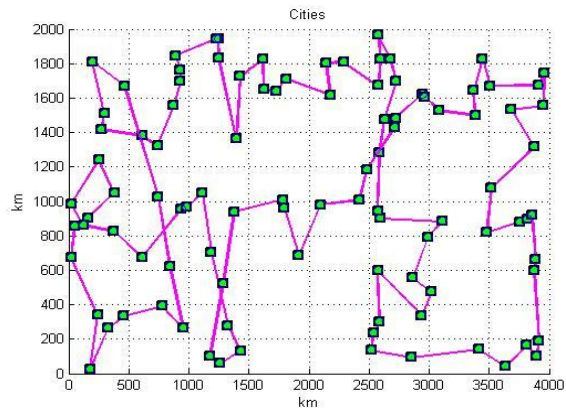
**Figure 4.29.** Best solution tour of Kro100 by WOA.



**Figure 4.30.** Best solution tour of Kro100 by GOW.



**Figure 4.31.** Best solution tour of Kro100 by CSA.



**Figure 4.32.** Best solution tour of Kro100 by CSO.

Table 4.2 shows comparisons of the performance of WOA, GWO, CSA, CSO and PSO to other existing algorithms that are taken from scientific literatures: adapted harmony search algorithm, improved discrete bat algorithm, discrete penguins search optimization, discrete cat swarm optimization, hunting search algorithm, shuffled frog leaping algorithm, multi-population discrete firefly algorithm, African buffalo optimization, discrete bacterial memetic evolutionary algorithm, clonal selection algorithm, discrete artificial bee colony algorithm, biogeography migration algorithm, discrete swallow swarm optimization algorithm, hunting search algorithm.

**Table 4.2:** Best result values obtained by different meta-heuristic algorithms.

Method	Dataset (optimal solution)					
	Oliver30 (423)	Eil51 (426)	Berlin52 (7542)	St70 (675)	Eil76 (538)	KroA100 (21282)
HS(Bouzidi and Riffi, 2014)	-	426	-	675	538	21282
IBA(Osaba and et al., 2016)	420	426	7542	675	539	21282
PeSOA(Mzili and Riffi, 2015)	-	426	7542	675	538	21282
CSO(Bouzidi and Riffi, 2013)	-	426	7542	675	538	21282
HUS(Agharghor and Riffi, 2015)	-	426	7542	675	538	21282
OXIMSFLA(Saud and et al., 2018)	434	534	8362	892	733	37212
CXIMSFLA(Saud and et al., 2018)	556	671	12266	1355	1072	58069
MDFA(Zhou and et al., 2014)	-	432	7681	682	-	-
ABO(Odili and Mohmad Kahar, 2016)	-	426	7542	-	538	21311
TSPBMA (Mo and Xu, 2011)	-	-	-	-	-	21282
CLONALG(Pang and et al., 2015)	-	441	7752	-	565	-
CLONALG+2opt(Pang and et al., 2015)	-	433	7598	-	562	-
DABC(Meng and et al., 2016)	423	431	7680	-	-	-
DBMEA(Kóczy and et al., 2017)	-	-	7544	-	-	-
DSSO(Bouzidi and Riffi, 2017)	-	426	-	675	538	-
GWO	423	429	7680	684	569	21984
WOA	423	437	7661	679	569	21958
CSA	423	439	7742	697	571	22451
CSO	423	444	8092	776	582	24020
PSO	801	907	16144	2001	1662	114001

## 5. CONCLUSION AND RECOMMENDATIONS

### 5.1 Conclusion

According to the obtained results for the six TSP datasets we can clearly explain each algorithm depending on its performance.

- Grey Wolf Optimizer: This algorithm showed very satisfactory performance in solving TSP were the value of oliver30 was optimum and for the other datasets it was too close the optimum value which makes this algorithm better than CSA, CSO and PSO. SO the performance of the GWO in terms of exploration, exploitation, local optima avoidance, and convergence was very satisfactory because it depends on the social hierarchy and hunting behavior of grey wolves were all the wolf members (alpha, beta, delta and omega) change their position in the search space depending on one equation.
- Whale Optimization Algorithm: This algorithm also showed very satisfactory performance in solving TSP were the value of oliver30 was optimum and for the other datasets it was too close the optimum value which makes this algorithm better than CSA, CSO and PSO and comparing to GWO this algorithm works slightly in a better way. SO the performance of the GWO in terms of exploration, exploitation, local optima avoidance, and convergence was excellent because it depends on two maneuver were the whale members change their position in the search space depending on a randomly selected probability that makes them able to choose one of the two maneuver equations (upward-spiral equation or random position equation).
- Crow search algorithm: This algorithm have a good ability to solve TSP were the value of oliver30 was optimum and for the other datasets it was too close the optimum value which makes this algorithm better than CSO and PSO. This algorithm passes GWO and WOA in working time were it requires less time to make the operation due to its simplicity. SO the performance of the CSA in terms of exploration, exploitation, local optima avoidance, and convergence was satisfactory because it depends on awareness probability of crows were the crow members change their position in the search space depending on that probability which



makes them able to fool each other by using one of the two equations (exact position of followed crow equation or random position equation).

- **Chicken Swarm Optimization:** Comparing this algorithm to the previous ones we can say that it has intermediate performance in solving TSP even in the working time were it requires a big amount of time but it still better than PSO were it obtained good results. SO the performance of the CSO in terms of exploration, exploitation, local optima avoidance, and convergence was good enough because this algorithm depends on the hierarchal order of chicken swarms were the swarm should be grouped into three groups every ten generation and their index should be determined. Furthermore, there are random following between hens and roosters and between chicks and mother hens. Each group has a different equation to change the group member's position in the search space. All that processes lead the operation to be more complex and require more time.
- **Particle Swarm Optimization:** This old method was the worst one in solving TSP among all the others, but it was the fastest one because it has less operations. SO the performance of the PSO in terms of exploration, exploitation, local optima avoidance, and convergence was worst because PSO algorithm solves continues optimization problems. The improvements on this algorithm was not good enough to make it suitable for TSP were depending on the random values of  $\alpha$  and  $\beta$  means random influence of personal best position and global best position will be considered on velocity calculation selecting more SOs and then using the new velocity to change the particles position.

## 5.2 Recommendations

This study was about five meta-heuristic algorithms GWO, WOA, CSA, CSO and PSO to solve TSP. The obtained results were satisfactory for the algorithms but it can be improved in other ways. In the future these algorithms can be improved by using local search algorithms such 2-Opt local search algorithm, 3-Opt local search algorithm and lin-kernighan type exchange. Also these algorithms can be hybridized with other algorithms to work in better way such as GWO with PSO.

## REFERENCES

- Abbass, H. A., 2001, MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach, *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, 207-214.
- Agharghor, A. ve Riffi, M. E., 2015, Hunting Search Algorithm To Solve The Traveling Salesman Problem, *Journal of Theoretical & Applied Information Technology*, 74 (1).
- Askarzadeh, A., 2016, A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm, *Computers & Structures*, 169, 1-12.
- Banzhaf, W., 1990, The “molecular” traveling salesman, *Biological Cybernetics*, 64 (1), 7-14.
- Bellman, R. ve Dreyfus, S., 1962, *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ, T962.
- Bellman, R. E. ve Dreyfus, S. E., 2015, *Applied dynamic programming*, Princeton university press.
- Bock, F., 1958, An algorithm for solving travelling-salesman and related network optimization problems, *Operations research*, 897-897.
- Bookstaber, D., 1997, Simulated Annealing for Traveling Salesman Problem, *SAREPORT*. nb.
- Bouzidi, A. ve Riffi, M. E., 2013, Discrete cat swarm optimization to resolve the traveling salesman problem, *International Journal*, 3 (9).
- Bouzidi, M. ve Riffi, M. E., 2014, Adaptation Of The Harmony Search Algorithm To Solve The Travelling Salesman Problem, *Journal of Theoretical & Applied Information Technology*, 62 (1).
- Bouzidi, S. ve Riffi, M. E., 2017, Discrete swallow swarm optimization algorithm for travelling salesman problem, *Proceedings of the 2017 International Conference on Smart Digital Environment*, 80-84.
- Croes, G. A., 1958, A method for solving traveling-salesman problems, *Operations research*, 6 (6), 791-812.
- Dantzig, G., Fulkerson, R. ve Johnson, S., 1954, Solution of a large-scale traveling-salesman problem, *Journal of the operations research society of America*, 2 (4), 393-410.
- Dorigo, M. ve Gambardella, L. M., 1997, Ant colonies for the travelling salesman problem, *biosystems*, 43 (2), 73-81.
- Eberhart, R. ve Kennedy, J., 1995, A new optimizer using particle swarm theory, *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, 39-43.
- Flood, M. M., 1956, The traveling-salesman problem, *Operations research*, 4 (1), 61-75.
- Formato, R. A., 2007, Central force optimization, *Prog Electromagn Res*, 77, 425-491.
- Glover, F. ve Taillard, E., 1993, A user's guide to tabu search, *Annals of operations research*, 41 (1), 1-28.
- Goldbogen, J. A., Friedlaender, A. S., Calambokidis, J., Mckenna, M. F., Simon, M. ve Nowacek, D. P., 2013, Integrative approaches to the study of baleen whale diving behavior, feeding performance, and foraging ecology, *BioScience*, 63 (2), 90-100.
- Gutin, G. ve Punnen, A. P., 2006, *The traveling salesman problem and its variations*, Springer Science & Business Media, p.
- Hamilton, W. R., 2000, *The Mathematical Papers of Sir William Rowan Hamilton*, CUP Archive, p.

- Han, F., Ling, Q.-H. ve Huang, D.-S., 2010, An improved approximation approach incorporating particle swarm optimization and a priori information into neural networks, *Neural Computing and Applications*, 19 (2), 255-261.
- Helsgaun, K., 2000, An effective implementation of the Lin–Kernighan traveling salesman heuristic, *European Journal of Operational Research*, 126 (1), 106-130.
- Holland, J. H., 1992, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, p.
- Huang, D.-S. ve Du, J.-X., 2008, A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks, *IEEE Transactions on neural networks*, 19 (12), 2099-2115.
- Hunt, J. E. ve Cooke, D. E., 1996, Learning using an artificial immune system, *Journal of network and computer applications*, 19 (2), 189-212.
- Kashan, A. H., 2009, League championship algorithm: a new algorithm for numerical function optimization, *2009 International Conference of Soft Computing and Pattern Recognition*, 43-48.
- Kashan, A. H., 2011, An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA), *Computer-Aided Design*, 43 (12), 1769-1792.
- Kaveh, A. ve Farhoudi, N., 2013, A new optimization method: dolphin echolocation, *Advances in Engineering Software*, 59, 53-70.
- Kóczy, L. T., Földesi, P. ve Tüü-Szabó, B., 2017, An effective discrete bacterial memetic evolutionary algorithm for the traveling salesman problem, *International Journal of Intelligent Systems*, 32 (8), 862-876.
- Kumbharana, S. N. ve Pandey, G. M., 2013, Solving travelling salesman problem using firefly algorithm, *International Journal for Research in science & advanced Technologies*, 2 (2), 53-57.
- Lawler, E. L. ve Wood, D. E., 1966, Branch-and-bound methods: A survey, *Operations research*, 14 (4), 699-719.
- Lawler, E. L., 1985, *The traveling salesman problem: a guided tour of combinatorial optimization*, Wiley-Interscience Series in Discrete Mathematics.
- Li, M. D., Zhao, H., Weng, X. W. ve Han, T., 2016, A novel nature-inspired algorithm for optimization: Virus colony search, *Advances in Engineering Software*, 92, 65-88.
- Lin, L. ve Gen, M., 2009, Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation, *Soft Computing*, 13 (2), 157-168.
- Lin, S., 1965, Computer solutions of the traveling salesman problem, *Bell System Technical Journal*, 44 (10), 2245-2269.
- Lin, S. ve Kernighan, B. W., 1973, An effective heuristic algorithm for the traveling-salesman problem, *Operations research*, 21 (2), 498-516.
- Liu, X. ve Xiu, C., 2007, A novel hysteretic chaotic neural network and its applications, *Neurocomputing*, 70 (13-15), 2561-2565.
- Mahalanobis, P. C., 1940, A sample survey of the acreage under jute in Bengal, *Sankhyā: The Indian Journal of Statistics*, 511-530.
- Mech, L. D., 1999, Alpha status, dominance, and division of labor in wolf packs, *Canadian Journal of Zoology*, 77 (8), 1196-1203.

- Meng, L., Yin, S. ve Hu, X., 2016, A new method used for traveling salesman problem based on discrete artificial bee colony algorithm, *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 14 (1), 342-348.
- Meng, X., Liu, Y., Gao, X. ve Zhang, H., 2014, A new bio-inspired algorithm: chicken swarm optimization, *International conference in swarm intelligence*, 86-94.
- Menger, K., 1932, Das botenproblem, *Ergebnisse eines mathematischen kolloquiums*, 2, 11-12.
- Merz, P. ve Freisleben, B., 1997, Genetic local search for the TSP: New results, *Evolutionary Computation, 1997., IEEE International Conference on*, 159-164.
- Mirjalili, S., Mirjalili, S. M. ve Lewis, A., 2014, Grey wolf optimizer, *Advances in engineering software*, 69, 46-61.
- Mirjalili, S. ve Lewis, A., 2016, The whale optimization algorithm, *Advances in Engineering Software*, 95, 51-67.
- Mitchell, J. E., 2002, Branch-and-cut algorithms for combinatorial optimization problems, *Handbook of applied optimization*, 65-77.
- Mo, H. ve Xu, L., 2011, Biogeography migration algorithm for traveling salesman problem, *International Journal of Intelligent Computing and Cybernetics*, 4 (3), 311-330.
- Moosavian, N. ve Roodsari, B. K., 2014a, Soccer league competition algorithm, a new method for solving systems of nonlinear equations, *Int. J. Intell. Sci*, 4 (1), 7-16.
- Moosavian, N. ve Roodsari, B. K., 2014b, Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks, *Swarm and Evolutionary Computation*, 17, 14-24.
- Moscato, P., 1989, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, *Caltech concurrent computation program, C3P Report*, 826, 1989.
- Mucherino, A. ve Seref, O., 2007, Monkey search: a novel metaheuristic search for global optimization, *AIP conference proceedings*, 162-173.
- Muro, C., Escobedo, R., Spector, L. ve Coppinger, R., 2011, Wolf-pack (*Canis lupus*) hunting strategies emerge from simple rules in computational simulations, *Behavioural processes*, 88 (3), 192-197.
- Mzili, I. ve Riffi, M. E., 2015, Discrete penguins search optimization algorithm to solve the traveling salesman problem, *Journal of Theoretical & Applied Information Technology*, 72 (3).
- Odili, J. B. ve Mohmad Kahar, M. N., 2016, Solving the traveling Salesman's problem using the African Buffalo optimization, *Computational intelligence and neuroscience*, 2016, 3.
- Oftadeh, R., Mahjoob, M. ve Shariatpanahi, M., 2010, A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search, *Computers & Mathematics with Applications*, 60 (7), 2087-2098.
- Olorunda, O. ve Engelbrecht, A. P., 2008, Measuring exploration/exploitation in particle swarms using swarm diversity, *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, 1128-1134.
- Osaba, E., Yang, X.-S., Diaz, F., Lopez-Garcia, P. ve Carballo, R., 2016, An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems, *Engineering Applications of Artificial Intelligence*, 48, 59-71.
- Pang, W., Wang, K., Wang, Y., Ou, G., Li, H. ve Huang, L., 2015, Clonal selection algorithm for solving permutation optimisation problems: a case study of travelling salesman problem,

- International Conference on Logistics Engineering, Management and Computer Science (LEMCS 2015)*. Atlantis Press.
- Prior, H., Schwarz, A. ve Güntürkün, O., 2008, Mirror-induced behavior in the magpie (*Pica pica*): evidence of self-recognition, *PLoS biology*, 6 (8), p. e202.
- Rao, R. V., Savsani, V. J. ve Vakharia, D., 2011, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, *Computer-Aided Design*, 43 (3), 303-315.
- Rao, R. V., Savsani, V. J. ve Vakharia, D., 2012, Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems, *Information sciences*, 183 (1), 1-15.
- Robinson, J., 1949, On the Hamiltonian game (a traveling salesman problem), *RAND PROJECT AIR FORCE ARLINGTON VA*.
- Samanlioglu, F., Ferrell Jr, W. G. ve Kurz, M. E., 2008, A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem, *Computers & Industrial Engineering*, 55 (2), 439-449.
- Saud, S., Kodaz, H. ve Babaoğlu, İ., 2018, Solving Travelling Salesman Problem by Using Optimization Algorithms, *KnE Social Sciences*, 3 (1), 17-32.
- Tao, G. ve Michalewicz, Z., 1998, Inver-over operator for the TSP, *International Conference on Parallel Problem Solving from Nature*, 803-812.
- Von Frisch, K., 1974, Decoding the language of the bee, *Science*, 185 (4152), 663-668.
- Wang, K.-P., Huang, L., Zhou, C.-G. ve Pang, W., 2003, Particle swarm optimization for traveling salesman problem, *Machine Learning and Cybernetics, 2003 International Conference on*, 1583-1585.
- Watkins, W. A. ve Schevill, W. E., 1979, Aerial observation of feeding behavior in four baleen whales: *Eubalaena glacialis*, *Balaenoptera borealis*, *Megaptera novaeangliae*, and *Balaenoptera physalus*, *Journal of Mammalogy*, 60 (1), 155-163.
- Webster, B. ve Bernhard, P. J., 2003, A local search optimization algorithm based on natural principles of gravitation.
- Winston, W. L. ve Goldberg, J. B., 2004, *Operations research: applications and algorithms*, Thomson Brooks/Cole Belmont.
- Yang, J., Shi, X., Marchese, M. ve Liang, Y., 2008, An ant colony optimization method for generalized TSP problem, *Progress in Natural Science*, 18 (11), 1417-1422.
- Yang, X.-S., 2009, Firefly algorithms for multimodal optimization, *International symposium on stochastic algorithms*, 169-178.
- Yang, X.-S., 2010, *Nature-inspired metaheuristic algorithms*, Luniver press.
- Yang, X.-S., 2011, Metaheuristic optimization, *Scholarpedia*, 6 (8), 11472.
- Zhou, L., Ding, L. ve Qiang, X., 2014, A multi-population discrete firefly algorithm to solve tsp, In: *Bio-Inspired Computing-Theories and Applications*, Eds: Springer, p. 648-653.
- Zhou, Y., Luo, Q., Chen, H., He, A. ve Wu, J., 2015, A discrete invasive weed optimization algorithm for solving traveling salesman problem, *Neurocomputing*, 151, 1227-1236.

**INTERNET REFERENCES**

- URL1: <http://www-history.mcs.st-andrews.ac.uk/Biographies/Hamilton.html> (25-03-2018)
- URL2: <http://www-history.mcs.st-andrews.ac.uk/Biographies/Kirkman.html> (25-03-2018)
- URL3: <https://www.informs.org/Explore/History-of-O.R.-Excellence/Biographical-Profiles/Flood-Merrill-M> (25-03-2018)
- URL4: <http://www-history.mcs.st-and.ac.uk/Biographies/Whitney.html> (25-03-2018)
- URL5: <https://comopt.ifl.uni-heidelberg.de/> (25-03-2018)
- URL6: Smith, C.L., Zielinski, S.L.: The Startling Intelligence of the Common Chicken. Scientific American 310(2) (2014) [http://www.upc-online.org/thinking/140130\\_the\\_startling\\_intelligence\\_of\\_the\\_common\\_chicken.html](http://www.upc-online.org/thinking/140130_the_startling_intelligence_of_the_common_chicken.html) (17-04-2018)
- URL7: Grillo, R.: Chicken Behavior: An Overview of Recent Science <https://freefromharm.org/chicken-behavior-an-overview-of-recent-science/>(17-04-2018)
- URL8: Chicken, <http://www.poultryhub.org/production/husbandry-management/poultry-behaviour/> (17-04-2018)
- URL9: [https://animaldiversity.org/accounts/Corvus\\_corax/#behavior](https://animaldiversity.org/accounts/Corvus_corax/#behavior) (29-04-2018)
- URL10: <https://corvidresearch.blog/category/crow-behavior/> (29-04-2018)
- URL11: <http://news.bbc.co.uk/2/hi/science/nature/4286965.stm> (29-04-2018)
- URL12: <https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/> (05-09-2016)
- URL13: <https://www.seeker.com/migrating-birds-take-turns-leading-the-flock-1769477313.html> (27-03-2018)
- URL14: <https://www.slideshare.net/abdonajmeldin/chicken-swarm-optimization-cso/>(18-04-2018)
- URL15: [https://www.nature.com/scientificamerican/journal/v296/n4/box/scientificamerican0407-64\\_BX2.html](https://www.nature.com/scientificamerican/journal/v296/n4/box/scientificamerican0407-64_BX2.html) (01-10-2018)

**CV****KİŞİSEL BİLGİLER**

**Adı Soyadı** : Omar Mohammed Ahmed Ahmed  
**Uyruđu** : IRAQ  
**Dođum Yeri ve Tarihi** : Karkuk-iraq / 13.02.1990  
**Telefon** : 05334976658  
**Faks** :  
**e-mail** : omertotti@gmail.com

**EĐİTİM**

<b>Derece</b>	<b>Adı, İlçe, İl</b>	<b>Bitirme Yılı</b>
Lise	: Iraq High School	
Üniversite	: Eastern Mediterranean University	
Yüksek Lisans	: Selçuk University	
Doktora	:	

**İŞ DENEYİMLERİ**

<b>Yıl</b>	<b>Kurum</b>	<b>Görevi</b>
------------	--------------	---------------

**UZMANLIK ALANI:** Öğrenci

**YABANCI DİLLER:** Arapça-İngilizce

**BELİRTMEK İSTEĐİNİZ DİĐER ÖZELLİKLER**

**YAYINLAR:** Ahmed, O , Kahramanlı, H . (2018). Meta-Heuristic Solution Approaches for Traveling Salesperson Problem. International Journal of Applied Mathematics, Electronics and Computers, 6 (3), 21-26.