ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

A STUDY OF VANTAGE POINT NEIGHBOURHOOD SEARCH IN THE BEES ALGORITHM FOR COMBINATORIAL OPTIMIZATION PROBLEMS

M.Sc. THESIS

Sultan ZEYBEK

Department of Mathematical Engineering

Thesis Advisor: Prof. Dr. Kamil ORUÇOĞLU

MAY 2014

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

A STUDY OF VANTAGE POINT NEIGHBOURHOOD SEARCH IN THE BEES ALGORITHM FOR COMBINATORIAL OPTIMIZATION PROBLEMS

M.Sc. THESIS

Sultan ZEYBEK

(509111080)

Department of Mathematical Engineering

Mathematical Engineering Programme

Thesis Advisor: Prof. Dr. Kamil ORUÇOĞLU

MAY 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

BAKIŞ NOKTASI KOMŞULUK ARAMASININ ARI ALGORİTMASI İLE KOMBİNATORYAL OPTİMİZASYON PROBLEMLERİNE UYGULANMASI

YÜKSEK LİSANS TEZİ

Sultan ZEYBEK

(509111080)

Matematik Mühendisliği Anabilim Dalı

Matematik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Kamil ORUÇOĞLU

MAYIS 2014

Sultan Zeybek, a **M.Sc.** student of ITU **Institute of Science and Technology** student ID **509111080,** successfully defended the **thesis** entitled "**A STUDY OF VANTAGE POINT NEIGHBOURHOOD SEARCH IN THE BEES ALGORITHM FOR COMBINATORIAL OPTIMIZATION PROBLEMS**", which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Kamil ORUÇOĞLU** Istanbul Technical University	.....
Jury Members :	**Assoc. Prof. Dr. Ali ERCENGİZ** Istanbul Technical University	.....
	Assist. Prof. Dr. Ebubekir KOÇ Fatih Sultan Mehmet Vakıf University	.....

Date of Submission : 5 May 2014

Date of Defense : 28 May 2014

To my family,

FOREWORD

I would like to express my gratitude to all those who have helped me, in any way, to successfully complete my M.Sc. Thesis.

In particular, I would like to express my sincere thanks to my supervisor, Assistant Professor Ebubekir Koç, who introduced me the to the the charming world of the bees. Special thanks to him, for his encouragement, invaluable advice and guidance throughout my research and his huge support in our weekly meetings.

As a matter of fact without assistance of Assistant Professor Ebubekir Koç this thesis might not be possible to produce.

Finally, I would also like to thank my family for the support and encouragement they have given me always.

May 2014 Sultan ZEYBEK

(Research Assistant)

TABLE OF CONTENTS

Page

ABBREVIATIONS

SI : Swarm Intelligence
CO : Combinatorial Optimization
GA : Genetic Algorithms
EAs : Evolutionary Algorithms
ACO : Ant Colony Optimisation
PSO : Particle Swarm Optimization
SOAs : Swarm-based Optimisation Algorithms
BCO : Bee Colony Optimization
BA : Bees Algorithm
VBA : Virtual Bee Algorithm
ABC : Artificial Bee Colony
VPBA : Vantage Point Bees Algorithm
TSP : Travelling Salesman Problem
Vp-tree : Vantage Point Tree
n : Number of scout bees
m : Number of patches selected out of n visited points
e : Number of best patches out of m selected patches
nep : Number of bees recruited for e best patches
nsp : Number of bees recruited for the other (m-e) selected patches
ngh : Size of patches
sc : Shrinking constant
sat : Site abandonment threshold

LIST OF TABLES

Page

LIST OF FIGURES

Page

A STUDY OF VANTAGE POINT NEIGHBOURHOOD SEARCH IN THE BEES ALGORITHM FOR COMBINATORIAL OPTIMIZATION PROBLEMS

SUMMARY

The overall aim of this work is to prove the hypothesis that the new Vantage Point Bees Algorithm capable of solving combinatorial optimization problems.

In this thesis, Bees Algorithm is presented for Traveling Salesman Problem (TSP) with new neighbourhood local search algorithm. The Bees Algorithm for discrete problems including local and global search strategies used for algorithm. A new neighbourhood procedure was developed to deal with local search with combinatorial domains.

Chapter 2 introduces to basic concepts of the additional background material needed for the reader to fully understand the main body of this thesis. It also defines the notion of optimization and the combinatorial, continuous, and mixed-variable optimization problems using the same common framework. It reviews the background literature on optimization, the definition of optimization and optimization techniques and combinatorial optimization problems.

Chapter 3 reviews the definition of swarm intelligence and highlights swarm behaviours. Swarm Intelligence (SI) is an engineering branch and it is defined as the collective problem solving capabilities of social animals. There are lots of swarm-based optimization algorithms that mimic nature's methods to drive a search towards the optimal solution. The developments of population-based algorithms are also presented in this chapter and background literature on swarm-based optimisation algorithms relevant to the work presented. This covers the Genetic Algorithms (GAs), Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO) and bees-inspired algorithms including the Bees Algorithm itself. Behaviours of honey-bees in their natural environment, including food foraging are explained in details. Computational simulations of honey-bee behaviours are reviewed to show the link between nature and optimisation algorithms. Honeybees inspired algorithms are a branch of Swarm Intelligence algorithms, which are motivated by the fascinating behaviour of honeybees. Their behaviour is studied in order to develop metaheuristic algorithms that can mimic the bees searching abilities.

Chapter 4 describes a study of the main characteristics of the standard Bees Algorithm. This is undertaken through an exploration of the parameters of the algorithm in order to help understand the methods by which its performance is improved. Then, it focuses on enhancements to the Bees Algorithm for local and global search. The algorithm is improved with the addition of dynamic recruitment, proportional patch shrinking and site abandonment ideas.

The Bees Algorithm required six parameters. There are number of scout bees (n), number of selected sites (m), number of top-ranking (elite) sites among the m selected sites (e), number of bees recruited for each non-elite site (nsp), number of bees recruited for each elite site (nep), and neighbourhood size (ngh) and the stopping criterion. The algorithm starts with the n scout bees being placed randomly in the search space and presents a neighbourhood search associated with a random search. The Bees Algorithm involves global and neighbourhood search. In step 1 the algorithm starts with the n scout bees being placed randomly in the search space. In step 2 the fitnesses of the points visited by the scout bees are evaluated. In step 4, bees that have the highest fitnesses are chosen as "selected bees" and those sites that have been visited will be chosen for neighbourhood search. Then, in steps 5 and 6, the algorithm conducts searches in the neighbourhood of the selected bees in terms of more bees for the e best bees.

Chapter 5 describes applications of the Bees Algorithm in combinatorial domains. The Bees Algorithm as described above is applicable to both combinatorial and functional optimisation problems so the performance of the Bees Algorithm was tested on continuous and combinatorial problems. Travelling Salesman Problem (TSP) definition is given and several local search algorithms are suggested for the algorithm as well as site abandonment in continuous optimization problems.

In chapter 6 The Bees Algorithm with Vantage Point Neihbourhood Search is described. The simplest vp-tree construction begins with selecting a pivot element, vantage point randomly. Given a set S of metric space elements (i.e combinatorial search space elements), the algorithm returns pointer to the root of an optimized vp-tree that satisfied the local optimum value (for example in TSP returns the optimal tour for each iteration of recruit phase). The algorithm of making vantage point neighbourhood searh for the Bees Algorithm presents a modification of the neighbourhood search procedure in the Bees Algorithm for combinatorial domains. We proposed vantage point neighbourhood search procedure for the Bees Algorithm local search in the recruitment selection. To develop a new local search in the recruitment phase we use the vantage point tree algorithm with median calculations. The Bees Algorithm with vantage point neihbourhood procedure is suggested as an addition to the Bees Algorithm to deal with combinatorial domains. The algorithm is applied to the Travelling Salesman Problem (TSP) to show that the algorithm is both robust and efficient.

BAKIŞ NOKTASI KOMŞULUK ARAMASININ ARI ALGORİTMASI İLE KOMBİNATORYAL OPTİMİZASYON PROBLEMLERİNE UYGULANMASI

ÖZET

Bu tez çalışmasının temel amacı arıların kaynak arama davranışlarını modelleyen arı algoritmasının, kombinatoryal uzaylarda komşuluk arama fazına yeni bir yaklaşım geliştirilmesidir. Geliştirilen yaklaşım Gezgin Satıcı Problemine uygulanarak Gezgin Satıcı Problemi çözümünün en iyilenmesi amaçlanmıştır.

Bu tez altı bölümden oluşmaktadır ve birinci bölümde tezin amaç ve hedeflerinden bahsedilerek hipotezin açıklaması yapılmıştır. Tezin amacı sürü zekasına dayalı olarak geliştirilmiş sezgisel optimizasyon algoritmalarından biri olan arı algoritmalarının kesikli uzaylardaki lokal komşuluk aramasına yeni bir yaklaşım geliştirmektir.

Çalışmanın ikinci bölümü optimizasyon kavramının açıklanması ve optimizasyon problemlerinin karakterizasyonuna ayrılmıştır. Temel tanım ve teoremlerden bahsedilerek kombinatoryal optimizasyon problemlerinin matematiksel modellemesi açıklanmıştır. Optimizasyon metotları temelde kesin çözüm üreten klasik teknikler ve yaklaşık çözüm üreten modern sezgisel teknikler olmak üzere iki kategoriye ayrılmaktadır. Gerçek hayattaki optimizasyon problemlerinin birçoğu matematiksel formül geliştirilerek çözülemeyecek kadar karmaşıktır. Klasik yöntemlerle böyle bir problem çözülmeye çalışıldığında, çözüm çok uzun sürebilir ve uzun sürmesine rağmen istenilen sonuca ulaşılamayabilir. Bu şekilde tanımlanması kolay fakat çözümü oldukça karmaşık olan NP-zor optimizasyon problemlerine çözüm aranırken sezgisel (heuristic) yöntemler geliştirilmiştir. Klasik yöntemler probleme özgüdür ve genellikle amaç fonksiyonu ve kısıtların türüne (doğrusal, doğrusal olmayan vb.) ve modellemede kullanılan değişkenlerin türüne (tamsayı, gerçek sayı vb.) bağlıdır. Sezgisel algoritmalar ise genel amaçlıdır ve değişik gerçek dünya problemlerine uygulanabilir. Günümüzde karmaşık optimizasyon problemlerinin modellenmesi ve çözülmesinde doğal benzetimlerin kullanımı oldukça artmıştır. Özellikle büyük boyutlu kombinatoryal, tamsayılı ve doğrusal olmayan matematiksel problemlerin çözülmesinde klasik optimizasyon teknikleri yetersiz kaldığından sezgisel ve doğal fenomenlerden esinlenilerek algoritmalar geliştirilmektedir.

Kombinatoryal optimizasyon problemleri kesikli çözüm uzayına sahip problemler için en iyi çözümü arayan, dikkate alınan amaç fonksiyonunu en iyileyen kesikli karar değişkenlerinin değerlerini bulmayla uğraşan optimizasyon problemleridir. Kombinatoryal optimizasyon problemleri en azlama (minimization) ve en çoklama (maximization) olarak ikiye ayrılır. En kısa yol problemi, gezgin satıcı problemi, atama problemi, atölye çizelgeleme problemleri ve araç rotalama problemleri

kombinatoryal optimizasyon problemlerinin literatürdeki birçok uygulama alanlarından bazılarıdır.

Üçüncü bölümde sürü zekası ve sürü zekası yaklaşımını temel alarak geliştirilmiş sürü tabanlı sezgisel optimizasyon algoritmaları incelenmiştir. Sürü, birbirleriyle etkileşen dağınık yapılı bireyler yığını anlamında kullanılır. Bireyler insan, karınca veya arı olarak ifade edilebilir. Sürü zekâsı N adet temsilcinin bir amaca yönelik davranışı gerçekleştirmek ve hedefe ulaşmak için birlikte çalışması olarak ifade edilmektedir. Doğadan esinlenen algoritmaların yeni bir dalı olan sürü zekâsı yaklaşımı, canlıların içgüdüsel problem çözme becerilerini kullanan etkili metasezgisel yöntemler geliştirebilmek için canlı davranışlarını matematiksel olarak modellemeye odaklanmıştır. Canlılar arasındaki etkileşimin bir sonucu olan kolektif zekânın en önemli parçalarından biri ise bireysel böcekler arasındaki bilgi paylaşımıdır. Kolaylıkla gözlenebilen bu "kollektif zekâ" temsilciler arasında sık tekrarlanan davranışlardan doğmaktadır. Temsilciler faaliyetlerini idare etmek için basit bireysel kurallar kullanmakta ve grubun kalan kısmıyla etkileşim yolu ile sürü amaçlarına ulaşmaktadır. Grup faaliyetlerinin toplamından bir çeşit kendini örgütleme doğmaktadır. Kuş sürülerinin havada süzülmesi ve farklı şekiller alması, karıncaların yiyecek arama davranışları, balık sürülerinin beraberce yüzmesi, bal arılarının buldukları yiyecek kaynağının kalitesi hakkındaki bilgiyi paylaşmaları ve salınım dansı yapmaları bu sürü davranışlarından sadece birkaçıdır.

Sürü zekâsı (Swarm Intelligence) sürülerin davranışlarının nasıl modellenebileceğine ve sürüdeki bireyler arasındaki iletişimin mantığını çözmeye dair çalışmaları kapsamaktadır. Sürüde özerk yapıdaki basit bireyler kollektif bir zeka geliştirerek iletişim kurarlar ve birbirlerinin hareketlerini kendinden organizasyon yardımıyla (self-organization) önceden yapılmış herhangi bir plan olmadan yönlendirebilirler. Bu ise esnek ve sağlam, merkezi bir yönetim birimi olmadan yapılanmayı sağlar. Doğadaki bu sosyal sistemler sezgisel yöntemlerin geliştirilerek optimizasyon problemlerine uygulanmasıyla önemli tekniklerin ortaya çıkış noktası olmuştur.

Sezgisel yöntemlerin geliştirilmesinde kullanılan yöntemlerin başında son yıllarda literatüre kazandırılmış sürü zekası davranışlarını modelleyen algoritmalar gelmektedir. Modern sezgisel algoritmaların en temeli gelişeme dayalı algoritmalardır. Gelişime dayalı algoritmaların birçok çeşidi vardır ve genetik algoritmalar bu alanda literatürdeki problemlere uygulanmaktadır. Karınca sürülerinin koloni halindeki davranışlarını modelleyen Karınca Kolonisi Optimizasyonu, kuş veya balık sürülerinin sosyal davranışlarından esinlenerek geliştirilmiş Parçacık Sürü Optimizasyonu ve arıların yiyecek arama davranışlarını modelleyerek geliştirilmiş olan Arı Algoritmaları sürü tabanlı optimizasyon algoritmalarıdır.

Karınca Kolonisi Algoritmasında karınca çevre şartlarına göre besin kaynağı ile evi arasında gidebileceği yolları belirlemektedir. Belirlenen yollardan birinden ilk geçen karınca yola feremon adında kimyasal bir koku bırakmaktadır. Koku yoğunluğu zamana bağlı olarak azalmaktadır. Eğer yol kısa ise bu koku daha yoğun olmaktadır. İki yolun kesiştiği noktada karınca hangi yola gideceğini belirlemektedir. Hangi yolu seçeceğine ilk önce koku miktarının yoğunluğuna göre ikinci olarak ise gelişigüzel bir ölçüte göre karar vermektedir. Bu gelişigüzel seçimin nedeni ise bütün karıncaların aynı yolda gitmesini engelleyerek yeni ve daha kısa yolları keşfetmektir.

Sezgisel yöntemlerin bir diğeri olan Parçacık Sürü Optimizasyonu (PSO) tekniği ilk olarak kuş ve balık sürülerinin hareketlerinden esinlenerek doğrusal olmayan nümerik problemlere optimal sonuçlar bulmak için ortaya atılmıştır ve basitleştirilmiş sosyal sistemin bir simülasyonu olarak ortaya çıkmıştır.

Arıların yiyecek arama davranışı, bilgi paylaşımı ve ezberleme özellikleri, son zamanlarda sürü zekâsında en ilginç araştırma alanlarından biri olmuştur. Kaliteli bir yiyecek kaynağı bulan arılar, yiyecek kaynağı hakkındaki yön, uzaklık ve nektar miktarı bilgilerini dans aracılığıyla diğer arılarla paylaşır. Bu başarılı mekânizma sayesinde arı kolonisi kaliteli yiyecek kaynağının olduğu bölgelere yönlendirilir.

Dördüncü bölüm Arı Algoritmalarının temel teorisini, işleme mekanizmasını ve literatürdeki uygulamalarını kapsamaktadır. Arı Algoritması (AA) ilk olarak Pham (2005) tarafından önerilmiş olup, bal arılarının yiyecek arama davranışını taklit eden popülasyon tabanlı bir arama algoritmasıdır.

Temel Arı Algoritması birçok parametre içermektedir: izci arı sayısı (n), ziyaret edilen n nokta içinden seçilen bölge sayısı (m), seçilen m bölge içindeki en iyi bölge sayısı (e), en iyi e bölgeye gönderilen arı sayısı (nep), kalan (m-e) bölgeye gönderilen arı sayısı (nsp), bölge boyutu (ngh) ve durdurma kriteri. Algoritma n adet izci arının araştırma uzayına rastgele yerleştirilmesi ile başlar. İzci arılarca ziyaret edilen noktaların uygunlukları 2. adımda değerlendirilir. 4. adımda en iyi uygunluk değerine sahip arılar elit arılar olarak, bu arılara ait bölgeler de komşuluk araması için seçilir. 5 ve 6. adımlarda seçilen arıların komşuluğunda araştırma başlar ve daha umut verici çözümleri temsil eden en iyi e bölgeye, seçilen diğer bölgelere göre daha fazla arı gönderilerek daha detaylı arama yapılır. 7. adımda yeni popülasyonun oluşturulması için her bölgedeki en iyi uygunluk değerine sahip arı seçilir. 8 nolu adımda popülasyondaki diğer arılar (*n*-*m*) yeni potansiyel çözümler elde etmek için rastgele olarak araştırma uzayına atanırlar. Her bir iterasyonun sonunda yeni popülasyon iki parçadan oluşacaktır: seçilen her bir bölgenin temsilcileri ve rastgele arama yapan izci arılar (Pham, 2006a, 2006b). Algoritma durdurma kriteri sağlanana kadar devam ettirilir. Burada 4-7 arası adımlar temel Arı Algoritmasının lokal arama (recruitment phase, local search) kısmını, son adım ise global arama kısmını oluşturmaktadır. Global arama ile rastgelelik (randomness) şansı devam ettireilerek algoritmanın olası yeni çöümler keşfetmesini sağlamak amaçlanmaktadır.

Beşinci bölümde kombinatoryal uzaylarda Arı Algoritmasının uygulanması incelenmiştir. Gezgin Satıcı Problemi gibi NP-zor kombinatoryal optimizasyon problemleri Arı Algoritması ile en iyilenirken Arı Algoritmasının komşuluk arama fazında çeşitli modifikasyonlara gidilmektedir.

Gezgin Satıcı Problemi, aralarındaki uzaklıkları bilinen noktalardan bir kez geçmek şartı ile tüm noktaların en az maliyetle dolaşılıp, başlangıç noktasına tekrar dönülmesini amaçlayan kombinatoryal bir optimizasyon problemidir. Problemin meşhur olmasının ve araştırmacıların yoğun ilgisini çekmesinin sebebi, gerçek hayattaki birçok soruna uyarlanabilir olmasıdır.

Farlı bir komşuluk aramasının Arı Algoritmasıyla birleştirilerek Gezgin Satıcı Problemine uygulanması ile bu tez çalışması tamamlanmıştır. Temel Arı Algoritması komşuluk yaklaşımındaki uzaklık fonksiyonun tanımından dolayı sürekli optimizasyon problemlerine rahatça uygulanabilirken, kombinatoryal uzaylarda komşuluk araması çeşitli komşuluk operatörleri kullanılarak yapılabilmektedir.

Altıncı bölümde bakış noktası (vantage point) komşuluk aramasının Arı Algoritması ile Gezgin Satıcı Problemine uygulaması yapılmıştır. Gezgin Satıcı Problemi metrik TSP olarak tanımlanmış olup Simetrik Gezgin Satıcı Problemi metrik uzay tanımıyla uygulamaya alınmıştır. Metrik uzay yapısını kullanan yöntemlerin arama algoritmaları özellikle yüksek boyutlu uzaylarda iyi performans sergilemektedir. En yakın k komşu taraması, benzerlik taraması metrik uzaylarda kullanılan arama algoritmalarındandır. Bakış noktası (Vantage Point) seçilerek herhangi bir metrik uzayda Vantage Point Tree oluşturulması metrik uzaylardaki arama algoritmalarının bir başka uygulamasıdır. Vantage Point Arı Algoritması sırasıyla rastgele seçilen bir yiyecek kaynağının diğer yiyecek kaynaklarına uzaklıklarının hesaplanmasıyla başlar. Daha sonra bulunan uzaklıkların medyanı hesaplanır. Medyandan küçük uzaklıkta bulunan yiyecek kaynakları ile medyandan büyük ya da medyana eşit olan olan uzaklıkta bulunan yiyecek kaynakları iki ayrı kümede toplanır. Optimum çözüme ulaşmak için medyandan küçük olan uzaklıktaki yiyecek kaynakları arasından yeni gidilecek olan yiyecek noktası rastgele olarak seçilir. Çözüm uzayındaki tüm noktalar, yani olası tüm yiyecek kaynakları (Gezgin Satıcı Problemi için gidilecek olan tüm şehirler) seçilene kadar iterasyona devam edilir. Bakış Noktası Arı Algortiması Temel Arı Algoritması ile aynı parametlere sahiptir. Uygulanan modifikasyon lokal arama fazında olup global arama kısmı Bakış Noktası Arı Algoritmasında aynı şekilde geçerlidir. Adımlar, durdurma kriteri sağlanana kadar devam ederek optimum sürede en iyi çözüme yakınsama amacına ulaşana kadar algoritma çalıştırılır. 51 şehirlik Gezgin Satıcı Problemine uygulanan bu algoritma kısa sürede optimal değere oldukça hızlı bir yakınsama gerçekleştirmiş.

1. INTRODUCTION

This thesis focuses on nature-inspired optimisation algorithms, in particular, the Bees Algorithm that developed for combinatorial domains with new local search procedure and applied to Traveller Salesman Problem (TSP).

The Travelling Salesman Problem (TSP) is one of the most interesting and challenging combinatorial optimization problems. TSP is all about finding a Hamiltonian path with minimum cost. It may be defined as a problem that is a simple to describe, but a difficult to solve, which is why it has received so much attention from the scientific community. This problem is a mathematical NP-hard problem and has a world range of applications for many fields such as transportation, logistics and semiconductor industries. (Karaboga and Gorkemli, 2011).

To solve the problem, many researchers have proposed different approaches including metaheuristic methods. Some animal behaviours have a potential to be adapted to solve TSP. In nature, there exist many processes which seek a stable state and these processes can be seen as natural optimization processes (Eberhart, 2001). Over the last 30 years, several attempts have been made to develop global optimization algorithms that simulate these natural optimization processes.

In the literature a lot of metaheuristic algorithms have been applied to optimization problems to obtain better results in reasonable computational times. Some of these algorithms include Evolutionary algorithms may be considered as one of the first of this class of algorithms (Koç, 2010). Other algorithms include Ant Colony Optimization (ACO) (Dorigo et al., 1996), Particle Swarm Optimization (Kennedy and Eberhart, 1995) and bees-inspired algorithms including the Bees Algorithm (Pham et al. 2005, 2006a).

In this thesis, Bees Algorithm is presented for Traveling Salesman Problem (TSP) with a new neighbourhood local search algorithm. The Bees Algorithm for combinatorial optimization problems including local and global search strategies

used for algorithm. A new neighbourhood procedure is developed to deal with local search with combinatorial domains.

1.1 Purpose of Thesis

In this thesis an efficient and robust local neighborhood search algorithm is proposed for combinatorial domains to increase the efficiency of the Bees Algorithm and it have been used successfully for the solution of the Travelling Salesman Problem (TSP).

Neighborhood search is vital constituent of all swarm based optimization algorithms. The Bees Algorithm has originally developed for continuous domains but combinatorial domains need a completely different approach when it comes to mathematical definition of the mathematical distance (Koç, 2010).

We aim to enhance the Bees Algorithm's neighbourhood search procedure defined for combinatorial domains and improve its performance for combinatorial domains as its performing for continuous domains.

For continuous domains in the original Bees Algorithm, "ngh" defines the initial size of neighbourhood for local searching. For example, if *x* is the position of an elite bee in the i^{th} dimension, follower bees will be replaced randomly in the interval $x_{ie} \pm ngh$ is set to define the boundaries of local search for new solution to improve the solution quality and performance (Ghanbarzadeh et al. 2007).

For combinatorial domains, combinations of several methods have been deployed to perform the neighborhood search. After modifying the neighbourhood part of Bees algorithm we compare it with several exchange neighbourhood strategies and local search algorithms including simple (2 point) swap, double (4 point) swap, insert, 3 point swap, 2-Opt and 3-Opt (Koç, 2010).

In this thesis, the performance of the Bees Algorithm optimization with vantage point local search algorithm is evaluated for the Travelling Salesman Problem (TSP) and the results are compared with the original Bees Algorithm including several exchange local search strategies.

In the context of developing an algorithm first, the biological and morphological features of honeybees are presented. Then original Bees Algorithm is presented with

vantage point local search. Then we are proposing mathematical simulation with experimental for understanding the succesfullnes of the modified algorithm.

1.2 Literature Review

The rapid development of engineering sciences and increases in the number of complex processes in industry and manufacturing mean that traditional optimisation techniques are no longer adequate to solve complex multi-variable optimisation problems with large numbers of parameters. These usually require intelligent optimisation tools such as the Bees Algorithm (Pham et al. 2005; 2006b).

Over the years, swarm intelligence has inspired scientist to developed population-based algorithms to deal with many complex multi-variable optimization problems. Because of many complex multi-variable optimization problems cannot be solved exactly within polynomially bounded computation times population-based algorithms were implemented (Koç, 2010). A recent trend in the field of Swarm Intelligence (Sl) is population-based algorithms and they are the utilisation of tools to solve optimisation problems which are defined as minimisation of cost functions.

Among the most common population-based algorithms are Evolutionary Algorithms (EA), the Genetic Algorithms (GA) (Goldberg, 1989), Particle Swarm Optimization (PSO) (Eberhart and Kennedy 1995), Ant Colony Optimization (ACO) (Dorigo et al., 1991; Dorigo et al., 1996) and bees-inspired algorithms including the Bees Algorithm (Pham et al., 2005, 2006) itself which mimics the foraging behaviour of honeybees in nature.

Evolutionary Algorithms (Rechenberg et al, 1965), (Fogel et al, 1966) and Genetic Algorithms may be considered as one of the first class of this class of algorithms. (Koç, 2010). The Genetic Algorithms (Holland, 1975) is based on biological evolution and adaptation in nature. Although they are considered in population-based algorithms, they may also separated from swarm-based optimization due to their centralised control mechanism. Particle Swarm Optimization algorithm imitates the action of flying, swimming or walking agents keeping themselves close by other members in a swarm. Ant Colony Optimisation (Dorigo et al., 1991), is inspired by the ants' foraging behaviour where they tend to choose the shortest route that links the food source and their nest which have no centralised control over their individuals.

In addition to these algorithms, The Bees Algorithm (Pham et al., 2005), which imitates the foraging behaviour of honey bees, is a bees-inspired algorithm. The algorithm has been widely applied to solve many complex optimisation problems and received a number of improvements (Ahmad, 2012). The Bees Algorithm is both implemented for continuous domains and combinatorial domains. For combinatorial domains, it is quite difficult to implement the current algorithm since it has been proposed originally for continuous domains. Therefore, it is interesting to explore the opportunities and limitations of the improved algorithm to this challenging new domain for the BA.

Four forms of honeybee behaviour have emerged in the literature, namely, the foraging behaviour (Seeley, 1996), the nesting site selection (Seeley and Visscher, 2003; Passino et al., 2008), the mating behaviour (Sung, 2003, Haddad et al., 2006) and the honeybee teamwork strategy (Sadik et al., 2006). These types of behaviour have been modelled to derive various Bees Algorithms with many applications (Otri, 2011).

The traveling salesman problem (TSP) is a well-known NP-hard optimization problem, in which we require to determine the shortest closed route passing through a set of n cities under the condition that each city is visited exactly once. Many problems in science, engineering, and bioinformatics fields, such as flexible manufacturing systems, routing as well as scheduling problems, physical mapping problems (Alizadeh, 1993), genome rearrangement (Sankoff, 1997) and phylogenetic tree construction (Korostensky, 2000), can be formulated as a TSP. A large number of approaches have been devoted to solve the TSP.

1.3 Hypothesis

The overall aim of this work was to prove the hypothesis that the Bees Algorithm with vantage point local search of neighbourhood is capable of solving Travelling Salesman Problem, which belongs to NP-hard optimization problem efficiently and robustly. We want to implement an efficient algorithm, which improves the local search structure of Bees Algorithm. In this thesis, Bees Algorithm is presented for Traveling Salesman Problem (TSP) with a new neighbourhood local search algorithm. The Bees Algorithm for combinatorial optimization problems including local and global search strategies used for algorithm. A new neighbourhood procedure is developed to deal with local search with combinatorial domains.

The objectives of this work were:

1. To implement a new local search algorithm for combinatorial domains to increase the efficiency of the Bees Algorithm.
2. To determine whether a vantage point local search neighbourhood improves the efficiency of the Bees Algorithm.
3. To compare both the original and improved versions of local search strategies of the Bees Algorithm for Travelling Salesman Problem.

2. MOTIVATIONS, BACKGROUND AND BASIC DEFINITIONS

This chapter provides o comprehensive inside into background to understand the main body of this thesis. It also defines the notion of optimization and the combinatorial, continuous, and mixed-variable optimization problems using the same common framework.

2.1 Optimization

Optimization is everywhere and is one of the most important tools in different fields of engineering (Yang, 2010). However, many optimization problems turns out to be very difficult and can not be solved exactly within a polynomially bounded computation times (Pham et., al 2006). The latest developments over the last two decedes tend to use metaheuristic optimization techniques to solve such NP-hard problem.

Definition 2.1 *Optimization* is the search for a set of variables that either maximize or minimize a scalar cost function, $f(\vec{x})$.

The n-dimensional decision vector, $\vec{x}$, consists of the n decision variables over which the decision maker has control. The cost function is multivariate since it depends on more than one decision variable, as is common of realworld relationships. The goal is to minimize (maximize) the cost function while satisfying the constraints in the problem.

2.2 Optimization Problems

An optimization problem defined as follows [Boyd and Vandenberghe 2004]:

Definition 2.2 Given a function $f : S \rightarrow \mathbb{R}$,find $s^* \in S : \forall x \in S\, f(s^*) \leq f(X)$ (minimization) or $f(s^*) \geq f(X)$ (maximization).

Function f is called objective function, its domain S is called the search space, and the elements of S**,** are called feasible solutions. A feasible solution X is a vector of optimization variables $X = \{X_1, X_2, \ldots, X_n\}$. A feasible solution X^* that minimizes (or maximizes) the objective function is called an optimal solution.

Each optimisation problem consists of four essential components:

1) An objective function or fitness function to be optimised,
2) A set of variables that need to be calculated to find the value of the objective function(s),
3) A set of constraints that determine the allowed values of the variables,
4) The search space that encompass all possible solutions to a problem.

With regards to these four components:

1. The degree of nonlinearity of the objective function determines whether the problem solved is a linear or nonlinear problem. In addition, if we try to classify optimization problems according to number of objectives, then there are two categories: in one objective function it is called a single-objective problem, otherwise, in a multi-objective problem a number of objective functions are needed.
2. The type of variables employed that divides problems into either continuous problems, or discrete and combinatorial problems, must be considered. In continuous problems the variables employed in the objective function are real values, whereas in discrete and combinatorial problems they are restricted to assume only discrete values (Socha, 2007) :
    - discrete optimization problems in which all the optimization variables X_i, $i = 1, \ldots, n$ are discrete, i.e., belong to a countable set, $X_i \in D_i, i = 1, \ldots, n.$
    - continuous optimization problems in which all the optimization variables X_i, $i = 1, \ldots, n$ are continuous $X_i \in \mathbb{R}$, $i = 1, \ldots, n$.
    - mixed-variable optimization problems in which p out of $n = p + q$ variables are discrete, $X_i \in D_i, i = 1, \ldots, p$ and q are continuous $X_i \in \mathbb{R}, i = p+1, \ldots, p+q$.

3. If the problem has no constraints or conditions that satisfy it, it is called an unconstrained problem, otherwise it is called a constrained problem where it contains one or more constraints that must be satisfied.
4. The search space determines if the problem is a static/deterministic problem which does not change over time, or if it is a dynamic/stochastic problem where the search space changes over time (Blackwell and Branke, 2004), (Otri, 2011).

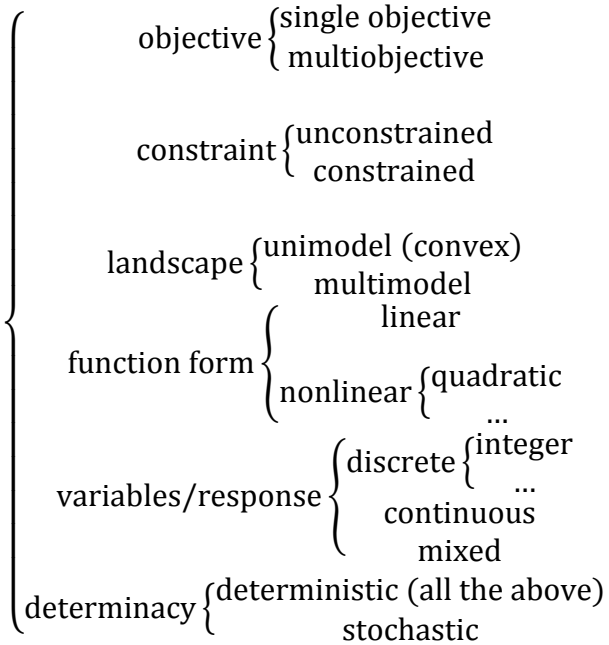


Figure 2.1 : Classification of optimization problems (Yang, 2010).

Optimization problems involving a large number of finite solutions often arise in academia, government, and industry. However, for many real-world optimization problems, it is not necessary to guarantee to find an optimal solution. Often a reasonably good (or approximate) solution is sufficient to find. Hence, optimization algorithms and approximate methods were born.

In this thesis, we mention Travelling Salesman Problem that plays an important role in combinatorial optimization problems. Combinatorial optimization problems are in fact a subset of *discrete* optimization problems characterised by finite size of their domain.

2.3 Combinatorial (Discrete) Optimization Problems

The name given to combinatorial optimization problems, i.e. combinatorial, comes from the fact that such problems may be expressed as those of finding a permutation or combination of a finite set of elements. Combinatorial optimization problems are therefore characterized by a finite set of possible solutions and it is a branch of optimisation in applied mathematics and computer science, related to operations research, algorithm theory and computational complexity theory.

Definition 2.3 A combinatorial optimization problem $\boldsymbol{P} = (\boldsymbol{S}, \boldsymbol{f})$ can be defined by $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables with domain $D_1 \ldots D_n$,and constraints with variables defined over subsets of S where an objective function,

$$f : D_1 \times D_2 \ldots . \times D_n \rightarrow \mathbb{R}^+$$

to be minimized and the set of all possible feasible assingment is

$$S = \left\{ s = \left\{ (x_1, v_1), \ldots, (x_n, v_n) \right\} \mid v_i \in D_i, \text{ s satisfies all the constraints} \right\}$$

S is solution space or search space of the optimization problem, as each element of set can be seen as a candidate solution but one has to find a solution $\boldsymbol{s} * \in \boldsymbol{S}$ with minimum objective function value, that is $f(s^*) \leq f(s), \ \forall \ \boldsymbol{s} \in \boldsymbol{S}$ and $\boldsymbol{s} * \in \boldsymbol{S}$ is called a globally optimal solution of $(\mathbf{S}, \mathbf{f})$.

Many algorithms and solution methods exist for solving combinatorial optimization problems. Some of them are exact methods called exact or complete algorithms that are guaranteed to find for optimal solutions given sufficient time called deterministic algorithms. Some others are approximation techniques, usually called metaheuristics, within stochastic algorithms which will give a good problem solution in a reasonable amount of time, with no guarantee to achieve optimality.

Algorithms often compared in terms of their efficiency, robustness and speed. Algorithm analysis is usually compared actual running time of algorithms and the *O* notation is often used to provide an asymptotic upper bound of the complexity of an algorithm. An algorithm is of *O*(n) (Order n), where *n* is the size of the problem, if the total number of steps carried out by the algorithm is at most a constant times *n*.

In addition to analyzing the efficiency of an algorithm, it is sometimes necessary to know what types of algorithms exist for solving a particular problem. The field of complexity analysis analyzes problems rather than algorithms. Two important classes of problems are usually identified in this context. The first class is called *P* (polynomial time problems). It contains problems that can be solved using algorithms with running times such as *O(n), O(log(n))* and $O(n^k)$ .They are relatively easy problems. Another important class is called *NP-hard* (non-deterministic polynomial time problems). NP-hard problems don't have a a polynomial-time solution, for example TSP is a well-known combinatorial NP-hard optimization problem (Hosny, 2010).

In this study, practical solutions for TSP problem are addressed. In the literature, there are useful theories, solutions, and case studies to solve such combinatorial optimization problems and an optimization problem can be solved using metaheuristic algorithms. That are mostly nature-inspired and population-based. Population based algorithms is about basic concept of swarm intelligence theory. Swarm intelligence has inspired scientists to develop population-based algorithms to deal with complex optimisation problems.

From a mathematical point of view, basic ingredients of a combinatorial optimization problem are: an instance; for example in the TSP the set of cities and the set of costs of traveling; a finite space of *feasible solutions* in the TSP, all the possible round-trips with requested properties and a *cost function* over the space of feasible solutions in the TSP, the total cost of every round-trip. The optimization problem is solved when, given an instance, a feasible solution which minimizes the cost function is found. Swarm Intelligence (SI) and popular swarm-based algorithms in several optimization tasks and research problems and it have been successfully applied in a variety of problem domains.

3. SWARM INTELLIGENCE

3.1 Swarm Intelligence

Swarm Intelligence (SI) is an engineering branch and it is defined as the collective problem solving capabilities of social animals (Bonabeau , 1999), (Koç, 2010). There are variety of the interesting insect or animal behaviour in the nature, for example a flock of birds sweeps across the sky. A group of ants forages for food, a school of fish swims, turns, flees together etc. Scientists call this kind of aggregate motion “swarm behavior” and they studied how to model biological swarms to understand how such social animals interact, achieve goals, and evolve.

Figure 3.1 : Fish scooling (left) and Birds flocking in V-formation (Xiong, 2010).

Swarm Intelligence is the emergent collective intelligence of groups of simple autonomous agents (Bonabeau, 1999). An autonomous agents is a subsystem that interacts with its environment, which probably consist of other agents but acts independent from all others agents (Liu, 2000).

SI is the direct result of self-organisation in which the interactions of lower-level components create a global-level dynamic structure that may be regarded as intelligence (Koç, 2010). These lower level interactions are guided by a simple set of rules. Individuals of colony only have local-level information about environment and they follow without any knowledge of global effects (Dorigo, 1999).

There are several other optimization techniques based on SI principles have been proposed in the literature, including Artificial Bee Colony (Karaboğa, 2005), Bacterial Foraging (Passino, 2002), Ant Colony Optimization (Dorigo, 1999), Artificial Immune System (De Castro, 1999) and Glowworm Swarm Optimization (Krishnanand and Ghose 2009). All these SI models intrinsically share the principal inspirational origin of the intelligence of different swarms in nature, such as swarms of E. coli bacteria as in Bacterial Foraging, swarms of cells and molecules as in Artificial Immune System (Hunt, 1996), (De Castro, 2002), (Read, 2012) and the amazing swarms of honeybees as in the Artificial Bee Colony System (Madureira, 2005), (Panigrahi, 2011).

3.2 Self-Organization in Nature

Self-Organization is a set of dynamical mechanism whereby structures appear at the global level of a system from interactions of its lower-level components. The four basis of self-organization are positive feedback (amplification), negative feedback (for counter-balance and stabilization), amplification of fluctuations (ramdomness, errors, random walks) and multiple interactions (Dorigo, 1999).

Positive feedback is defined as the first rule of self-organization and it is basically a set of simple rules that help to generate the complex structure. Negative feedback reduces the effects of positive feedback. Ramdomness adds an uncertainty factor to the system and enables the colonies to discover new solutions for example most challenging food sources, nest sites etc. Multible interactions between individulas are the last one. There should be minimum number of individulas individuals who are capable of interacting with each other to turn their independent local-level activities into one interconnected living organism. As a result of combination of these elements, a decentralised structure is created (Koç, 2010).

Usually there is no central control structure dictating how the individual agents should behave, but local interactions between such agents often lead to the emergence of a global behavior. Examples of systems like this can be found in nature, including ant colonies, bird flocking, bee swarming, animal herding, bacteria molding and fish schooling.

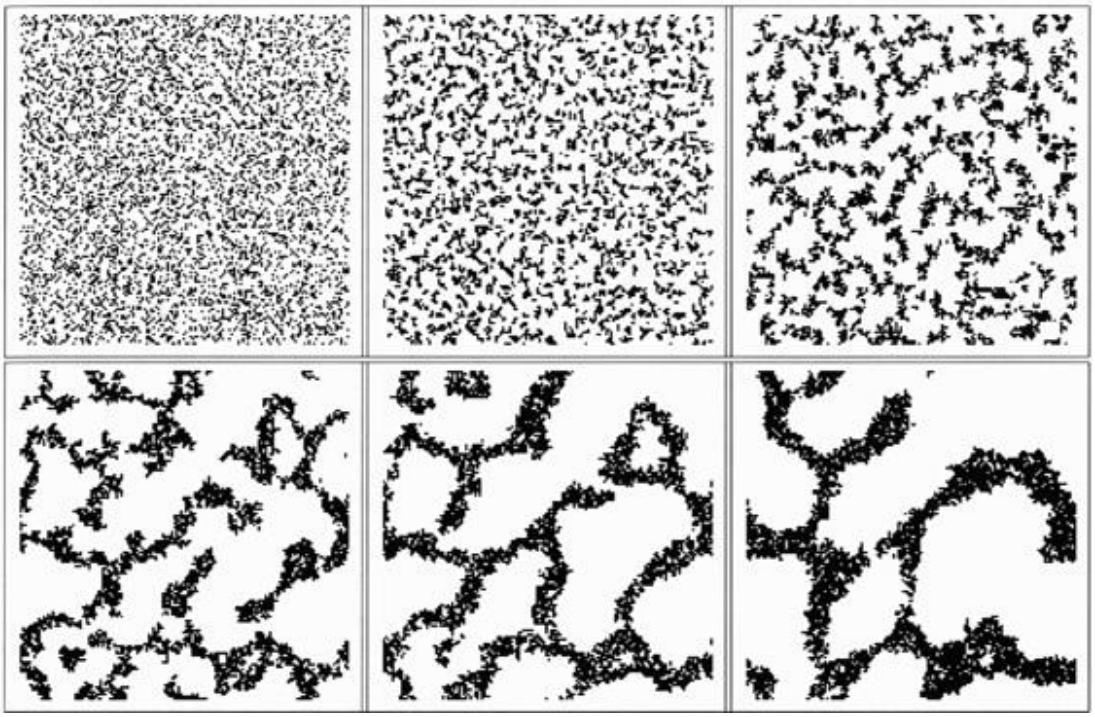


Figure 3.2 : Self-organization in a termite simulation (Mitchel Resnick, 1994).

3.3 Swarm-Based Optimization Algorithms

There are lots of swarm-based optimization algorithms (SOAs) that mimic nature's methods to drive a search towards the optimal solution. SOAs use a population of solutions for every iteration instead of a single solution (Koç, 2010). This is key difference between SOAs and the others types of search algorithms.

On the types of searches applied to solve the optimisation problem there are two possibilities: Single Point Search (Trajectory) (SPS) which is also known as a Direct Search (DS), and Population-Based Search (PBS) which is also known as a Swarm Based Search (SBS) (Otri, 2011).

SOAs include Evolutionary Algorithms (i.e. the Genetic Algorithm), the Ant Colony Optimisation (ACO) and the Particle Swarm Optimisation (PSO). In this section, we will focus on the main chararcteristics and the ways that each algorithm generate new solutions.

3.3.1 Genetic algorithms

Genetic Algorithms (GAs) was introduced by John Holland (Goldberg, 1989) and developed based on the genetic processes of biological organisms. It is based on natural selection and genetic recombination. It is a heuristic algorithm which simulates principles of evolution biology for finding solutions of complex problems which cannot be solved with any other exact algorithms. Genetic algorithms differ from the more normal optimization and search procedures in four ways (Goldberg, 1989) :

- GAs work with a coding of the parameter set, not the paramaters themselves;
- GAs search from a population of points, not a single point;
- GAs use ogjective function information, not derivatives or other auxiliary knowledge;
- GAs use probabilistic transition rules, not deterministic rules.

A genetic algorithm mimics this natural evolutionary process in its optimization problem cycle. A simple genetic algorithm based optimizer is characterized by individual encoding, individual fitness, selection mechanism and genetic operators.

Individual encoding means that genetic algorithms encode solutions to the given problem as chromosomal strings and operate on these encodings during the optimization process. This helps minimize the amount of problem specific information needed during the optimization process of a genetic algorithm. An encoding scheme that maps each chromosome string to a unique solution is preferred as the genetic algorithm will not waste time evaluating multiple encodings of the same solution. The solution is traditionally represented by binary numbers, string of zeros and ones but it is possible to use any other representation. For example, for a traveling salesman problem (TSP), a permutation of all the cities in the problem instance can be used a solution encoding scheme.

The fitness measure of the chromosome should reflect the quality of the corresponding solution to the problem. For example, in a TSP instance, the length of the overall tour represented by the permutation encoding could be assigned as the fitness measure.

The selection mechanism is used to select two individuals for crossover (mating). The purpose of these operations is to allow substrings in the fit individuals in a population to survive for many generations. Hence, the parent individuals for these operations are generally selected based on their fitness values. This will promote survival of fitter genes in the offspring and should lead to fitter individuals in the future generations.

Genetic algorithms use two kinds of genetic operators called crossover and mutation. The crossover operator performs a probabilistic exchange of chromosomal information between two individuals to produce a new individual. The crossover operator selects two parent individuals from the population based on a selection scheme.

The mutation operator typically picks a random individual from the population and performs an inversion or some other random operation on the individual chromosome. After a certain number of generations, the crossover operator tends to produce offspring that are very similar to the parent individuals. Then the mutation operator plays a critical role in restoring lost genetic material or providing diversity in the current population. *Initialization* – made first population which is usually generated randomly. This population can have any size – from a few to millions

1. *Evaluation* – each population is evaluated = there is computed so-called fitness function of given solution. The purpose of this function is to find out to how extent this solution fulfills given requirements. This requirement can have different form – the fastest computing as possible, the best solution as possible etc.
2. *Selection* – the purpose is to improve fitness value of population. So it is important to select just population which is the right pattern for find the best solution ≥ principle of evolution, only the strongest individuals can life. There are many methods of selection but the basic idea is still the same – selection of the best candidates for making the best possible future generation
3. *Crossover* – this operation makes new population by making hybrid of two selected populations – they can be called *parents*. The basic idea is to combine the best attributes of each parent.
4. *Mutation* – this operation makes the procedure of making new generation little bit random. It is important for possible improvement.
5. *Repeat!* - generate new generation and continue from step two.

Figure 3.3 : Pseudo-code of the GA algorithm.

Genetic Algorithms usually ends after a given count of iterations, after a given time of solving or after achieve given solutions. The solution of GAs heavily depends on defined on count of populations and count of iterations.

Count of iterations means how many times is algorithm repeated. First run has as input parameter random value but every next starts from the best founded solution from previous iterations. Count of iterations means how many times is algorithm repeated. First run has as input parameter random value but every next starts from the best founded solution from previous iteration.

3.3.2 Ant colony optimization algorithms

The successful swarm intelligence model is Ant Colony Optimization (ACO), which was introduced by M. Dorigo, and has been originally used to solve combinatorial optimization problems in the late 1980s. It is mimic the foraging behavior of social ant. It is a natural observation that a group of 'almost blind' ants can jointly figure out the shortest route between their food and their nest without any visual information.When searching for food, ants initially explore the area surrounding their nest randomly. As soon as an ant finds a food source, it evaluates the quantity and carries some of it back to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which depends on the quantity and quality of the food, will guide other ants to the food source.

Ant System (AS) was the first ACO approach to be published and it is an iterative distributed algorithm. (Dorigo et al., 1991; Dorigo et al., 1996), (Koç,2010). At each iteration, a set of artificial ants are considered. The general structure of any ACO algorithm, Starting with an initialization of the algorithm, iteration after iteration all ants first construct their tour and then update the pheromone trails accordingly. In an extended scenario i.e. an optional case, a local search method can be used to improve the ants' tours before updating the pheromone is positive feedback and decay is negative feedback. Pheromone is updated by all the ants after a complete tour is the key idea in this algorithm. Pheromone update (τ_{ij}) for the edges of the graph (a_j) that is joining the cities i and j is calculated as follows (Dorigo et al., 1991):

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta \tau_{ij}{}^{k} \qquad \textbf{(3.1)}$$

where m is the number of ants, $\rho \in (0,1]$ is the evaporation rate, and $\tau_{ij}{}^{k}$ is the quantity of pheromone laid on the edge (ij).

The value of the quantity of pheromone laid on the edges is determined by the tour length (L_k) of an ant defined by:

$$\tau_{ij}{}^{k} = \begin{cases} \dfrac{1}{(L_k)} & if\ ant\ k\ used\ edge(i,j)\ in\ its\ tour, \\ 0 & otherwise, \end{cases} \qquad \textbf{(3.2)}$$

Ants move from one city to another city according to probability. A transition functionis used to calculate the probability of an ant moving from the city i to j. Secondly, define a visible degree $n_{ij}, n_{ij} = 1/d_{ij}$. The probability of the kth ant choosing city is given by:

$$p(c_i j \,|\, s_k^p) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\, n_{ij}^{\beta}}{\sum_{c_{ij} \in N(s_k^p)} \tau_{il}^{\alpha} n_{il}^{\beta}} & if\ j \in N(s_k^p), \\ 0 & otherwise, \end{cases} \qquad \textbf{(3.3)}$$

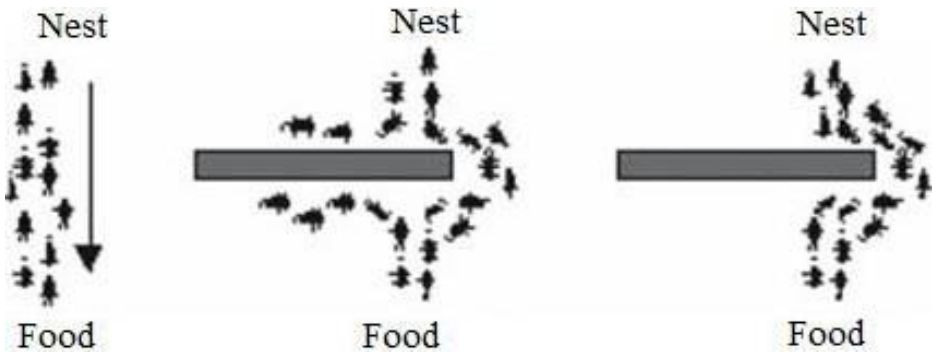


Figure 3.4 : Ants behaviour in finding the shortest route (Dorigo, 1996)

The basic idea and procedure of ACO algorithm:

1. Represent the solution space by a construction graph.

2. Set ACO parameters and initialize pheromone trails

3. Generate ant solutions from each ant‘s walk on the construction graph mediated by pheromone trails.

4. Update pheromone intensities.

5. Go to step 3, and repeat until convergence or termination conditions are met.

Figure 3.5 : Pseudo-code of the ACO algorithm.

There are hundreds of implementations of the ACO metaheuristic successfully applied to numerous optimization problems in various domains, including famous NP-hard combinatorial optimization problems. While ACO was initially introduced with an application to the TSP as a proof-of-concept/classical application, and then ACO algorithms have later been successfully applied to a wide-range of optimization problems.

3.3.3 Particle swarm optimization (PSO)

Particle Swarm Optimisation (PSO) was introduced by Kennedy and Eberhart (Eberhart and Kennedy 1995, Kennedy et al. 1997). PSO is population-based stochastic optimisation technique and is inspired by the behaviour of a flock of birds. The algorithm consists of a swarm of particles moving in a space. Every particle holds a position and velocity vector representing a candidate solution to the problem. In addition, each particle memorises its own best position found so far and a global best position that is obtained through communication with its neighbour.

Similar to evolutionary algorithms, the PSO initialises with a population of random solutions and it searches for local optima by simply updating generations of individuals. The pseudo-code o f the PSO algorithm is given below:

1. Create particles (population) distributed over solution space (s_i^0, v_i^0).

2. While (Stopping criterion not met) do

3. Evaluate each particle's position according to the objective function.

4. If S_i^{k+1} is better than S_i^k (update pbest)

$$S_i^k = S_i^{k+1}$$

5. Determine the best particle (update gbest).

6. Update particles' velocities according to

$$v_i^{k+1} = v_i^{kt} + c_1 rand_1\left(pbest_i - s_i^k\right) + c_2 rand_2(pbest - s_i^k)$$

7. Move particles to their new positions according to

$$s_i^{k+1} = s_i^k + v_i^{k+1}$$

8. Go to step 3, until stopping criteria are satisfied.

Figure 3.6 : Pseudo-code of the PSO algorithm (Eberhart and Kennedy, 1995).

The algorithm starts with creating particles that are uniformly distributed throughout the solution space by defining the initial conditions for each agent. Each agent is defined with an initial position (s_i^0) and an initial velocity (v_i^0). Each particle has a memory function that remembers two pieces of information, the first piece of information results from the memory of the particle of its past states as the best-so-far position that it has visited, called the local best, and the second piece of information results from the collective experience of all particles as the global best position attained by the whole swarm, called the global best. Both the local best position of each particle and the global best position of the entire swarm guide the movements of all particles towards new improved positions and eventuality to find the global minima/maxima (Otri, 2011).

3.3.4 Bees-inspired algorithms

3.3.4.1 Bees in nature

Honeybees inspired algorithms are a branch of Swarm Intelligence algorithms, which are motivated by the fascinating behaviour of honeybees. Their behaviour is studied in order to develop metaheuristic algorithms that can mimic the bees searching abilities in nature. There are several examples of such behaviour, such as waggle dance that is by a scout (worker) bees that has returned back to the comb with pollen or nectar. It is basically a language that "tells" other workers where the food is. By signaling both distance and direction with particular movements, the worker bee uses the dance language to recruit and/or direct other workers for gathering pollen and nectar.

Austrian ethologist Karl von Frisch was one of the first people to translate the meaning of the waggle dance (Frisch, 1967). Bees communicate through this waggle dance, which contains the following information:

- The direction of flower patches (angle between the sun and the patch)
- The distance from the hive (duration of the dance)
- The quality rating (fitness) (frequency of the dance)
- The order of the source by pollens on their legs (to specify the patch coordinates)

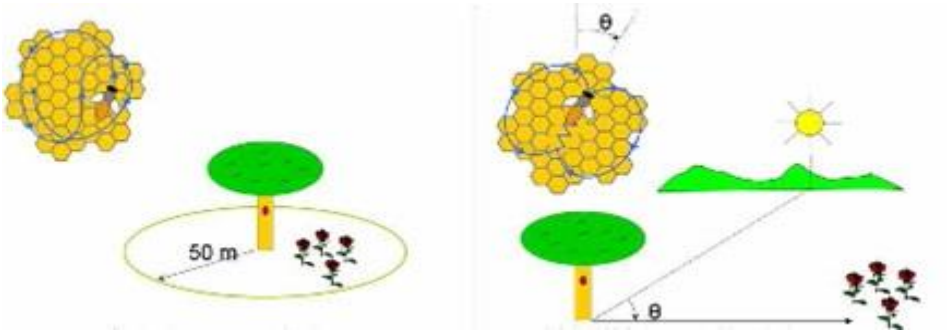


Figure 3.7 : Round Dance (left) and Waggle Dance of honeybees (Frisch, 1967).

Food sources that are at intermediate distances, between 50 and 150 meters from the hive, are described by the "sickle dance". This dance is crescent-shaped and

represents a transitional dance between the round dance and a waggle dance (Winston, 1987).

3.3.4.2 Nectar-source selection and the nest-site selection models

A colony of honey bees can extend itself over long distances (more than 10 km) and in multiple directions simultaneously to exploit a large number of food sources. Bees Algorithm starts with scout bees being placed randomly on the search space.

If the bees have no knowledge about the food sources in the search field they will be an unemployed foragers bees, so bee initializes its search as an unemployed. If the bee starts searching spontaneously without any knowledge, it will be a scout bee (Seeley, 1995). If the unemployed forager attends to a waggle dance done, the bee will start searching by using the knowledge from waggle dance is a recruit bee. When the recruit bee finds and exploits the food source, it will raise to be an employed forager who memorizes the location of the food source.

The value of a food source depends on different parameters such as its proximity to the nest, richness of energy and ease of extracting this energy. According to the fitness, patches can be visited by more bees or may be abandoned. The bees evaluate the different patches according to nectar quality and energy usage. By performing the waggle dance, successful foragers share the information about the direction and distance to patches of flower and the amount of nectar within this flower with their hive mates.

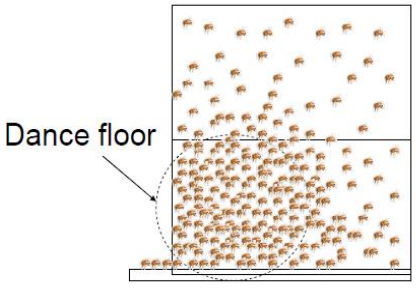


Figure 3.8 : The dancer bees meet other bees at the dance floor (Koç, 2010).

Camazine presented a differential equations model to honey bees' behaviors. Individual bees are represented in this model using a flow diagram for the nectar-source selection processes and they do not have global information about the

distribution of nectar sources each one will comply with certain rules to determine where it will go to forage. This process is described by a flow diagram illustrated in Fig. 3.7.

According to the model, there are seven decision making branches for the situation of a colony choosing between two nectar sources which nectar source to forage and whether to dance. There are foraging at nectar source A, foraging at nectar source B, dancing for nectar source D_A , dancing for nectar source D_B, unemployed foragers observing a dancer F, unloading nectar from source H_A and unloading nectar from source H_B .

In this model, there are two factors affecting the proportion of the total forager number in each compartment: (1) the rate at which a bee moves from one compartment to another and (2) the probability that a bee takes a fork at each of the five branch points (diamonds), r, stands for a rate constant defined as the fraction of bees leaving a compartment in a given time interval equal to $1/T$,, where each T, is the time to get from one compartment to another. The unit of the rate constant is given as min^{-1}(Camazine, 1991).

The first branch point is encountered after a bee has unloaded her nectar in the hive. Here, bee may abandon the nectar source and return to the dance floor to follow another dancer. P^X stands for the abandoning function that denotes the probability that a bee may abandon the nectar source or go back to the dance floor to observe another dancer bee. This function depends on the profitability of the source, so P_X^A represents the probability that a bee leaving H_A, abandoning the nectar source and becoming a follower bee (F).

The second branch point is for the bees that did not abandon their source. At this point, a bee decides whether to dance for the nectar source or to fly back to the nectar source. P_d denotes the probability of performing a dance for the nectar source. Its value also depends on the profitability o f thenectar source similar to the abandoning function P_d^A denotes the probability of performing a dance for the nectar source.

The third branch occurs on the dance floor when a follower bee dances to decide for one of the nectar sources P_F^A, denotes the probability of a follower bee following dances for nectar source A and leaving for this nectar source. Thus, the probability of

following a dancer bee for A (f_F^A), and the probability of following a dancer bee for B (f_F^B):

$$f_F^A = \frac{D_A d_A}{D_A d_A + D_B d_B} \quad \textbf{(3.4)}$$

$$f_F^B = \frac{D_B d_B}{D_A d_A + D_B d_B} \quad \textbf{(3.5)}$$

The time limitation of D_A and D_B has been weighted and denoted as d_A and d_B. Therefore, each function indicates the proportion of the total dancing for each nectar source by taking into account the number of dancers and the time spent dancing. Equations of the model, with some assumptions for simplicity, are written as the following set of differential equations (Camazine et al. 1991):

$$\frac{dA}{dt} = (1 - f_d^A)(1 - f_X^A) p_1 H_A + p_2 D_A + f_F^A p_4 F - p_3 A \quad \textbf{(3.6)}$$

$$\frac{dD_A}{dt} = f_d^A \left(1 - f_X^A\right) p_1 \mathrm{H_A} - p_2 \mathrm{D_A} \quad \textbf{(3.7)}$$

$$\frac{dH_A}{dt} = p_3 \mathrm{A} - p_1 \mathrm{H_A} \quad \textbf{(3.8)}$$

$$\frac{dB}{dt} = (1 - f_d^B)(1 - f_X^B) p_5 H_B + p_6 D_B + f_F^B p_4 F - p_7 A \quad \textbf{(3.9)}$$

$$\frac{dD_B}{dt} = f_d^B \left(1 - f_X^B\right) p_5 \mathrm{H_B} - p_6 \mathrm{D_B} \quad \textbf{(3.10)}$$

$$\frac{dH_B}{dt} = p_7 \mathrm{B} - p_5 \mathrm{H_B} \quad \textbf{(3.11)}$$

$$\frac{dF}{dt} = f_X^A p_1 H_A + f_X^B p_5 H_B - p_4 F \quad \textbf{(3.12)}$$

A detailed derivation and discussion of these equations is given in Camazine and Sneyd (1991).

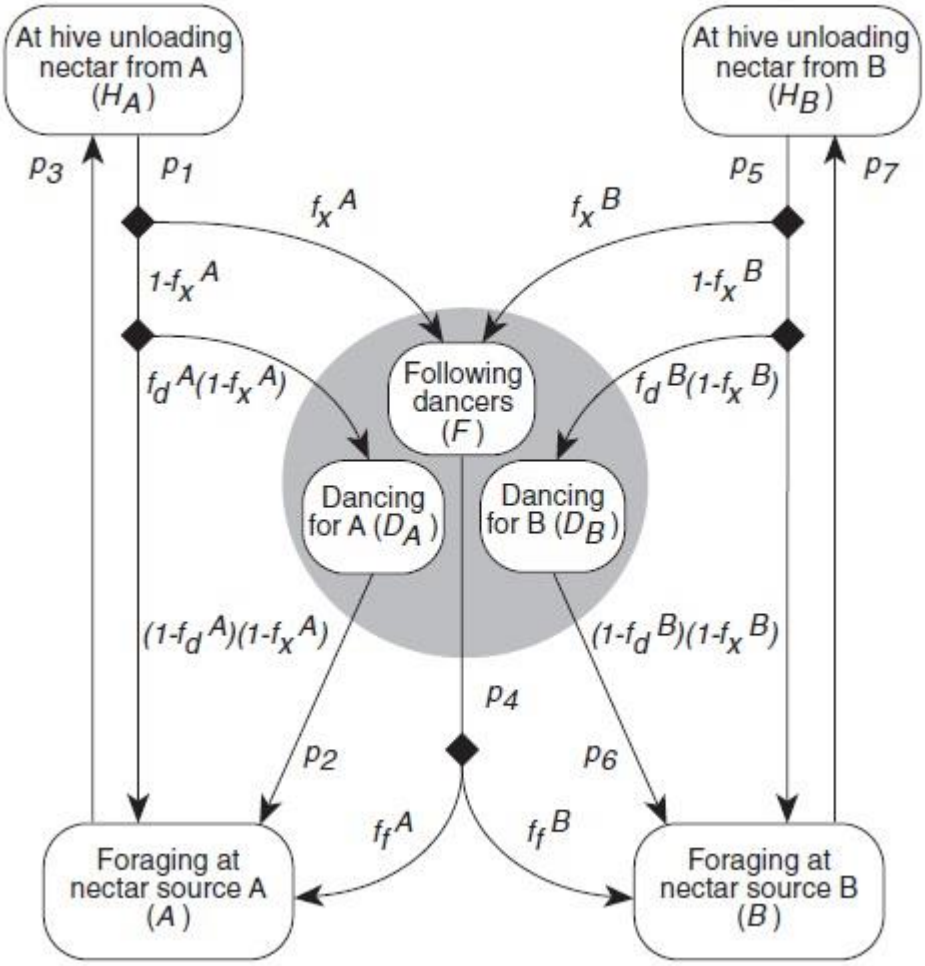


Figure 3.9 : A mathematical model shows how honey-bee colonies allocate foragers.

At any moment each forager can be in one of the seven compartments shown (H_A, H_B, D_A, D_B, A, B, F) denote the compartments as well as the number of foragers in the compartments). The rate at which bees leave each compartment is indicated by $p_1 - p_7$. The functions $f_X^A, f_X^B, f_d^A, f_d^B$ and so on, indicate the probability of taking one or the other fork at each of the five branch points black diamonds (Seeley Camazine, and Sneyd 1991) .

Nest-site selection is another important practice which requires an optimisation process as nectar source selection behaviour does in honey-bee colonies. Nest-site selection in honey-bee colonies can be summarised as a social decision making process. In this process, scout bees locate several potential nest sites, evaluate them, and select the best one on a competitive signalling basis (Passino and Seeley, 2006).

In nature, honey bees have several complicated behaviors such as mating, breeding and foraging and these behaviors have been mimicked for several honey bee based optimization algorithms. Honey bees optimization algorithms are categorized in this work by concerning the behavioural characteristics of honey bees. There are foraging behaviours, marriage behaviours and Queen bee concept.

The researches, their main contributions and applications are summarized as shown in Table3.1. Yonezawa and Kikuchi (1996) examine the foraging behaviour of honey bees and Sato and Hagiwara (1997) introduced honey-bees inspired algorithm called the bee system, as an improved version of genetic algorithms. This system claims to be inspired basically *from 'finding a source and recruiting others to i t '* behaviour.

Seeley and Buhrman (1999) investigated the nest site selection behaviour of honey bee colonies. The nest site selection process starts with several hundred scout bees. After the scouts return to the cluster, report their findings by means of waggle dances, and decide the new nest site. Luck and Teodorovic (2001) published the first study on Bee System based on the PhD thesis. They named the model the Bees System and aimed to deal with the Travelling Salesman Problem. So the algorithm was developed for combinatorial domains and applied to traveller salesman problems (TSP) that aim to find the minimum distance route.

Yang (2005) was inspired The Virtual Bee Algorithm (VBA) by a swarm of virtual bees where it began with bees wandering randomly in the search space. The VBA initially created a population of virtual bees, where each bee was associated with a memory bank. Then, the functions of optimisation (objectives) were converted into virtual food.

1. Creating a population of multi-agents or virtual bees, each bee is associated with a memory bank with several strings;
2. Encoding of the objectives or optimization functions and converting into the virtual food;
3. Defining a criterion fo r communicating the direction and distance in the similar fashion o f the fitness function or selection criterion in the genetic algorithms;

4. Marching or updating a population o f individuals to new positions fo r virtual food searching, marking food and the direction with virtual waggle dance;
5. After certain time o f evolution, the highest mode in the number of virtual bees or intensity/frequency of visiting bees corresponds to the best estimates;
6. Decoding the results to obtain the solution to the problem

Figure 3.10 : Pseudo-code of the VBA algorithm.

Karaboga and Basturk introduced the foraging behaviour of honey bee swarm and proposes a new algorithm simulating this behaviour for solving multi-dimensional and multi-modal optimization problems, called Artificial Bee Colony (ABC). The algorithm uses three types of bees, called employed bees, onlooker bees and scout bees and the main steps of the algorithm are:

1. Send the employed bees onto the food sources and determine their nectar amounts;
2. Calculate the probability value of the sources with which they are preferred by the onlooker bees;
3. Stop the exploitation process of the sources abandoned by the bees;
4. Send the scouts into the search area for discovering new food sources, randomly;
5. Memorize the best food source found so far.

For each flower patch, an artificial onlooker bee chooses a food source depending on the probability value associated with that food source, p_i, calculated by the following expression

$$p_i = \frac{fit_i}{\sum_{N=1}^{SN} fit_n} \qquad \textbf{(3.13)}$$

where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and SN is the number of food sourceswhich is equal to the number of employed bees or onlooker bee.

In order to produce a candidate food position from the old one in memory, the ABC uses the following expression:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \qquad \textbf{(3.14)}$$

where $k \in \{1; 2; \ldots; SN\}$ and $j \in \{1; 2; \ldots; D\}$ and $\varphi_{ij} \in [-1,1]$ are randomlychosen and $k \neq i$. ABC also uses site abandonment, which is simply leaving a patch if no more improvement is observed on the patch after certain number of iterations. It is called ‘‘limit”. Assume that the abandoned source is x_i and

$j \in \{1; 2; \ldots; D\}$ then the scout discovers a new food source to be replaced with x_i. It defined in the following equation:

$$x_j^i = x_{min}^i + rand[0,1](x_{max}^i - x_{min}^j) \quad \textbf{(3.15)}$$

1. Initialize the population of solutions x_i ; $\in \{1; \ldots; SN\}$
2. Evaluate the population
3. cycle = 1
4. **repeat**
5. Produce new solutions v_i for the employed bees by using (7) and evaluate them
6. Apply the greedy selection process for the employed bees
7. Calculate the probability values Pi for the solutions x_i by (6)
8. Produce the new solutions ti for the onlookers from the solutions xi selected depending on p_i and evaluate them
9. Apply the greedy selection process for the onlookers
10. Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution x_i by (8)
11. Memorize the best solution achieved so far
12. cycle = cycle + 1
13. until cycle = **MCN**

Figure 3.11 : Pseudo-code of the ABC algorithm. (Koç, 2010).

Another implementation of bee behaviour was presented by (Teodorovic, 2006) to solve transportation problems and was called Bee Colony Optimisation (BCO). A Fuzzy Bee System was also proposed in (Teodorovic et al., 2006). BCO has been developed for combinatorial problems and the pseudo-code of the algorithm is given below:

1. Initialization. Determine the number o f bees B, and the number of iterations I. Select the set of stages ST = { St_1, St_2,…,St_m}. Find any feasible solution *x* of the problem. This solution is the initial best solution.

2. Set i 1. Until i = I, repeat the following steps:

3. Set j 1. Until j = m, repeat the following steps:

Forward pass: Allow bees to fly from the hive and to choose B partial solutions from the set of partial solutionsS_j at stage st_j.

Backward pass: Send all bees back to the hive. Allow bees to exchange information about quality of the partial solutions created and to decide whether to abandon the created partial solution and become again uncommitted follower, continue to expand the same partial solution without recruiting the nestmates, or

dance and thus recruit the nestmates before returning to the created partial

solution. Set j: = j + 1.

4. If the best solution x_i obtained during the i-th iteration is better than the bestknown solution, update the best known solution (x: = x_i).

5. Set, i: = i + 1.

Figure 3.12 : Pseudo-code of the BCO algorithm (Koç, 2010).

4. THE BEES ALGORITHM

The Bees Algorithm was developed by a group of researchers at the Manufacturing Engineering Centre, Cardiff University (Pham et al., 2005). It is a population based search algorithm that mimics the food foraging behaviour of honeybees to find the optimal solution for both continuous and combinatorial problem. In its basic version, the algorithm performs a kind of neighbourhood search combined with random search.

The Bees Algorithm required six parameters. There are number of scout bees (n), number of selected sites (m), number of top-ranking (elite) sites among the m selected sites (e), number of bees recruited for each non-elite site (nsp), number of bees recruited for each elite site (nep), and neighbourhood size (ngh) and the stopping criterion. The algorithm starts with the n scout bees being placed randomly in the search space and presents a neighbourhood search associated with a random search.

The Bees Algorithm (BA) involves global and neighbourhood search. A number of bees are employed to explore at random the solution space in the global search procedure that enables the bees to escape from local optima. This kind of search is crucial as it enables the bees to escape from local optima. At the same time, neighbourhood search concentrates exploitation around promising solutions. Both of them in population-based algorithms may locate solutions that gradually come closer to an optimal. Initialize population with random solutions;

1. Evaluate fitness of the population.
2. While (stopping criterion not met)
3. Select sites for neighbourhood search.
4. Recruit bees for selected sites (more bees for e best sites) and evaluate fitnesses.
5. Select the fittest bee from each site to form the new population.
6. Assign remaining bees to search randomly and evaluate their fitnesses.
7. End While.

Figure 4.1 : Pseudo-code of the Bees Algorithm (Koç, 2010).

Table 4.1 : Basic parameters of the Bees Algorithm. (Koç, 2010).

Parameter	Symbols
Number of scout bees in the selected patches	n
Number of best patches in the selected patches	m
Number of elite patches in the selected best patches	e
Number of recruited bees in the elite patches	nep
Number of recruited bees in the non-elite best	nsp
The size of neighborhood for each patch	ngh
Number of iterations	Maxiter

In step 1 the algorithm starts with the n scout bees being placed randomly in the search space. In step 2 the fitnesses of the points visited by the scout bees are evaluated. In step 4, bees that have the highest fitnesses are chosen as "selected bees" and those sites that have been visited will be chosen for neighbourhood search. Then, in steps 5 and 6, the algorithm conducts searches in the neighbourhood of the selected bees in terms of more bees for the e best bees.

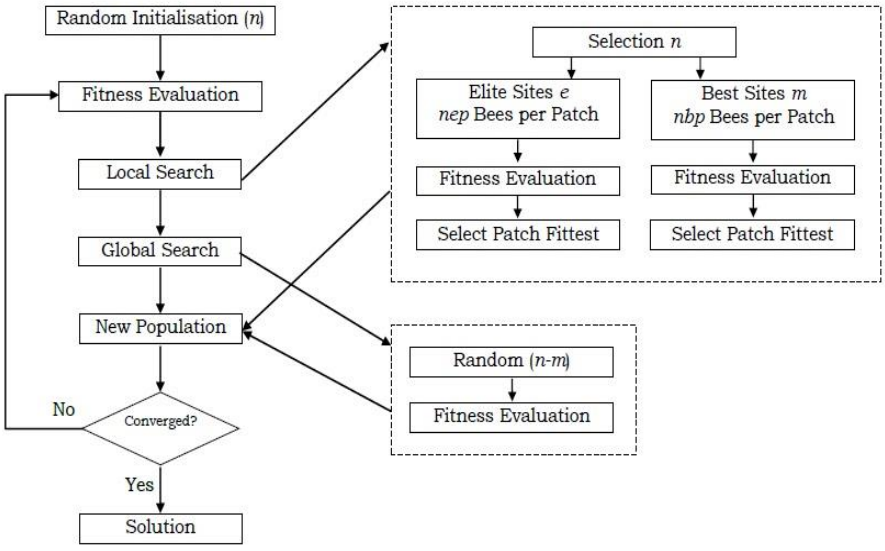


Figure 4.2 : Flowchart of the basic Bees Algorithm (Otri, 2011).

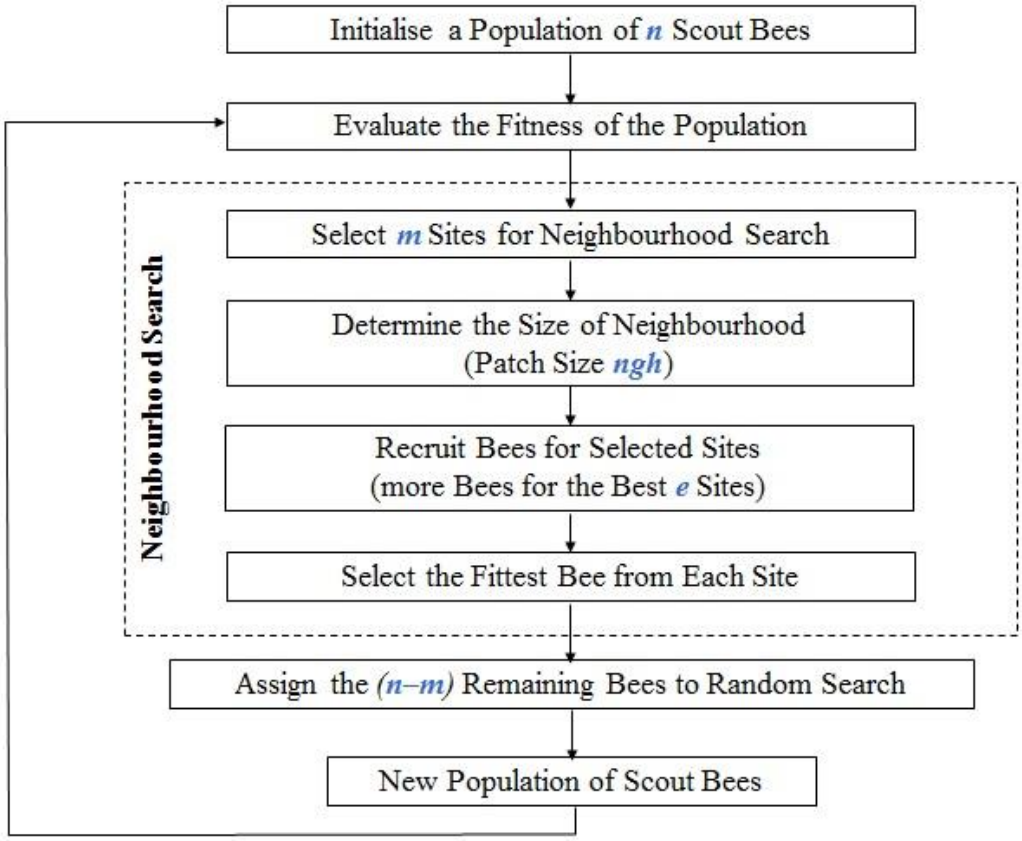


Figure 4.3 : Flowchart of the basic Bees Algorithm.

Example 4.1: Step1: Initialize population with random solutions with n = 20. Evaluate fitness of the population**.**

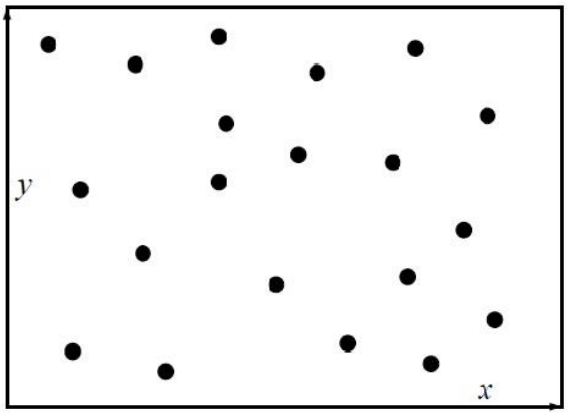


Figure 4.4 : 20 scout bees are placed randomly in the search space.

Step 2: Select the parameters of Bees Algorithm:

- ***n*** = 20 number of scout bees
- ***m*** = 3 number of sites selected out of *n* visited sites
- ***e*** = 1 number of best sites out of *m* selected sites
- ***nep*** = 7 number of bees recruited for best *e* sites
- ***nsp*** = 2 number of bees recruited for other (*m*-*e*) selected sites
- ***ngh*** = 3 neighbourhood size $x_{ie} \pm ngh$

Select sites for neighbourhood search. (Bees that have the highest fitnesses are chosen).

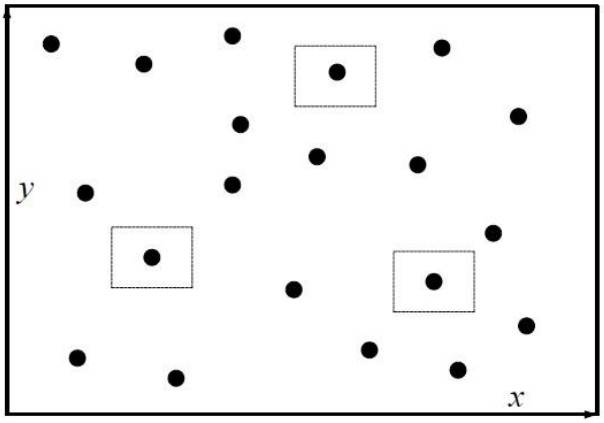
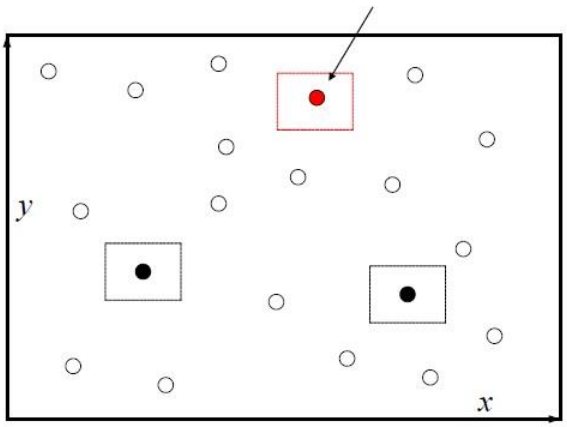


Figure 4.5 : m=3 selected bees for neighbourhood search.

Recruited bees for selected sites (more bees for best e sites) and evaluate fitnesses.

The best e sites (more bees for best e sites)



Select the fittest bee from each patch. (For each patch, only the bee with the highest fitness will be selected to form the next bees population.)

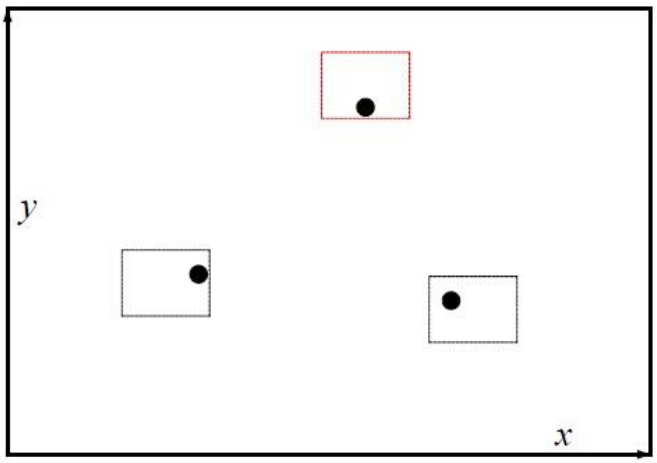


Figure 4.1 : Recruitment phase for local search.

Assing remaining bees to search randomly and evaluate their fitnesses.

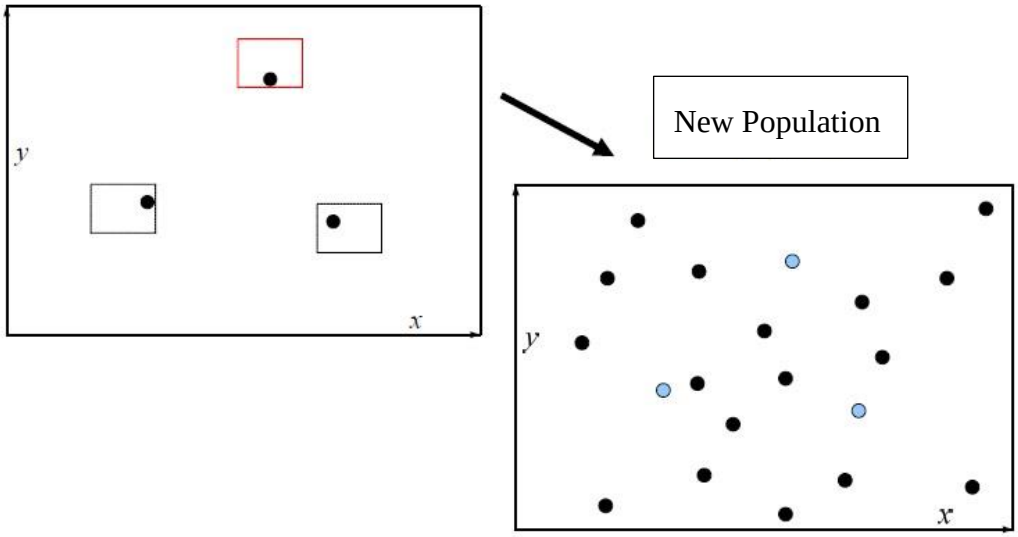


Figure 4.2 : Generate new population with local and global search phase.

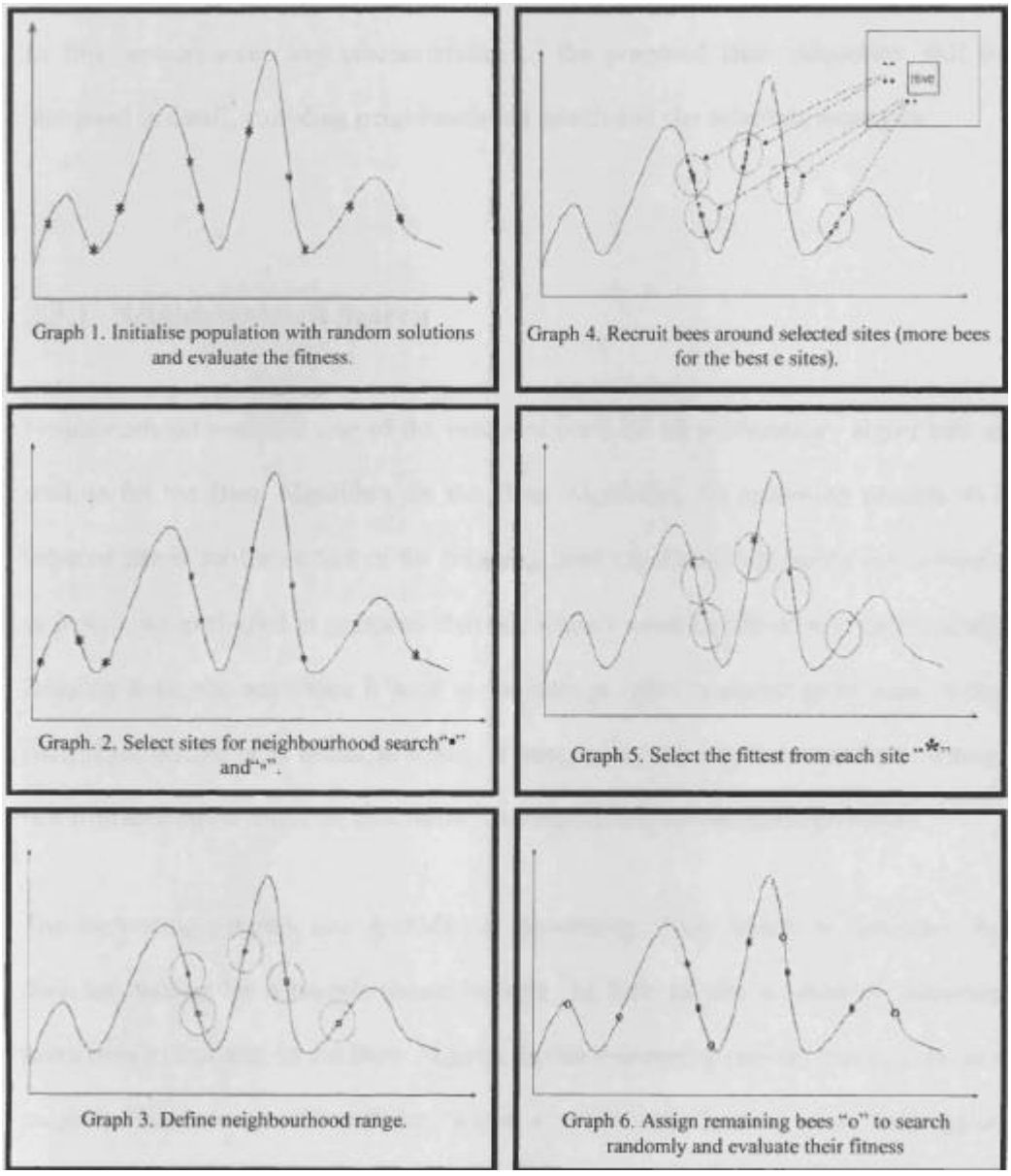


Figure 4.3 : Simple example of Bees Algorithm with n=10 scout bees (Koç, 2010).

4.1 Neighbourhood / Local Search of Bees Algorithm

As in all the evolutionary algorithms, the neighbourhood search is one of the essential parts of swarm-based algorithms as well as for the Bees Algorithm. In the Bees Algorithm, the searching process in a selected site is similar to that of the foraging field exploitation of honey bee colonies in nature. The harvesting process as explained in previous chapter includes a monitoring phase for the purpose of recruiting more bees to selected site that can be used as a neighbourhood search in

the Bees Algorithm. Essentially, when a scout bee finds a good field (good solution), she advertises her field to more bees. As we explained in the previous chapter the nesting-site selection of honeybees behaviour has been used as a neighbourhood search in the proposed Bees Algorithm.

After ranking the sampled solutions and locating the most promising ones (i.e. the highest ranking locations), other bees are recruited to search the fitness landscape in the neighbourhood of these solutions. A neighborhood search sites of size ngh is selected which will be used to update the m bees declared. This is important as there might be better solutions than the original solution in the neighborhood area.

In the neighbourhood search procedure, more forager bees are recruited in the neighbourhood of the elite (e) sites, and fewer bees around the non-elite (m-e) sites and thanks to this strategy the foraging effort was concentrated on the very best (i.e., elite) solutions. For every selected site, bees are randomly distributed to find a better solution within the given neighbourhood area (i.e., flower patch size). As shown in Fig. 4.7, only the fittest (best) bee is chosen as a representative bee and the centre of the neighbourhood shifted to the position of the best bee. (i.e from A to B).

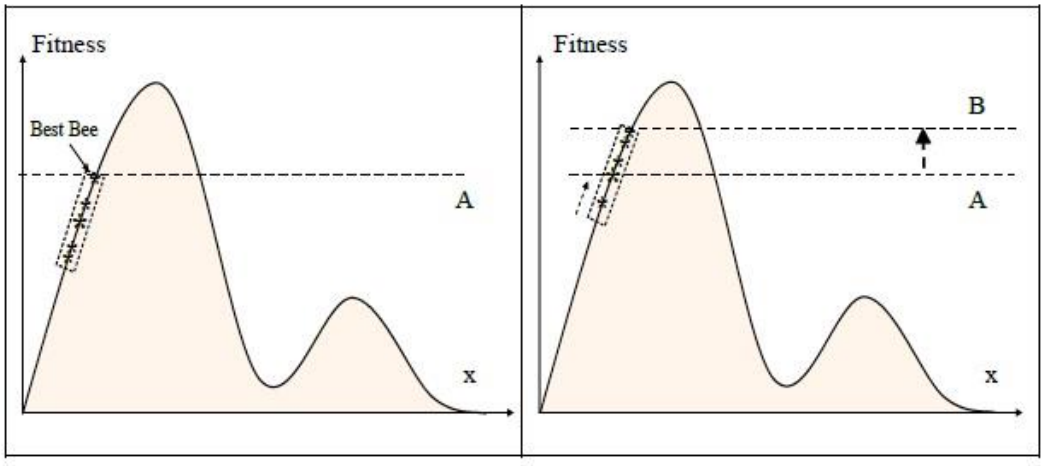


Figure 4.4 : Graphical Explanation of the Neighbourhood Search (Otri, 2011).

4.2 Improvements to the Bees Algorithm

We purpose to impove the efficiency of the Bees Algorithm in local search and global search with dynamic recruitment, proportional shrinking for selected sites and site abandonment.

Dynamic recruitment aim to improve the way that the bees are recruited into a selected site and it is deal with the local search space faster. With dynamic recruitment strategy if there is any improvement on the recruited site according to original bee, the recruited bee will replace the original and path will move to a new position around the fittest and new position.

Proportional shrinking idea defined with which the initial patch size is set as a starting patch size in the first iteration of the algorithm. Shrinking Constant (sc) is called the to a contraction of patch sizes of all selected sites (m) in every iteration of the algorithm proportional to a constant ratio. Depending on the iteration (i), the patch size of the site *m* ($\mathrm{Ngh_m(i)}$) is calculated as a contraction from the previous size ($\mathrm{Ngh_m(i-1)}$) proportional to the value of *sc*.

where $\mathrm{sc} \in [0,1]$ and $(\mathrm{Ngh_m(i)}) \geq 0$.

$$\mathrm{Ngh_m(i)} = \begin{cases} i = 1 & \mathrm{Ngh_m(i)} = \mathit{InitialPatchSize} \\ i > 1 & \mathrm{Ngh_m(i)} = \mathrm{Ngh_m(i-1)}\left(\left(1-\mathrm{SC}\right)\right) \mathit{and}\ \mathrm{Ngh_m}(i) > 0 \end{cases} \quad \textbf{(4.1)}$$

This strategy is proposed to improve solution quality and evoluation time. To improve the efficiency of local search we use site abondenment strategy. If there is no improvement of the fitness, value of the fittest bee after a certain number of iterations the site will be abondened.

The site abondenment strategy is proposed to escape from local in many complex optimization problems. Investigations are also given on details of the local and global search methods used in the algorithm. Also, details of the improvements made to local and global search methods are presented, including dynamic recruitment, proportional shrinking and abandonment strategies (Koç, 2010).

1. Initial population with n random solution.

2. Evaluate fitness of the population.

3. While (stopping criterion not met)

4. Select sites (m) for neighbourhood search.

5. Recruit bees for selected sites (more bees for best e sites), evaluate fitnesses, select the fittest bee from each site and shrink patches

for (k=l ; k=e ; k++) // Elite Sites

for (Bee=l ; Bee= nep ; Bee++) // More Bees for Elite Sites

BeesPositionlnNghO = GenerateRandomValueInNgh (*from x+ngh to x-ngh*),

Evaluate Fitness = Bee(i); //Evalute the fitnees of recruited Bee(i)

If (Bee(i) is better then Bee(i-l)) RepresentativeBee = Bee(i);

for (k=e ; k=m ; k++) // Other selected sites (m-e)

for (Bee=l ; Bee= nsp ; Bee++) // Less Bees for Other Selected Sites (m-e)

BeesPositionlnNghO = GenerateRandomValuelnNgh(*from x+ngh to x-ngh*);

Evaluate Fitness = Bee(i); //Evalute the fitnees o f recruited Bee(i)

If (Bee(i) is better then Bee(i-l))

RepresentativeBee = Bee(i);

6. for (patch=l; patch=m; patch++)

// Shrink all patches (m) proportional to SC

$\mathrm{Ngh_m(i)} = \mathrm{Ngh_m(i-1)}\big((1-\mathrm{SC})\big)$;

7. If (Iteration > sat)

If (no improvement on the site)

Save the Best Fitness;

Abandon the Site;

Bee(m) = GenerateRandomValue(All Search Space);

8. Assign remaining bees to search randomly and evaluate their fitnesses.

// (n-m) assigned to search randomly into whole solution space

Figure 4.5 : Pseudo-code of the improved Bees Algorithm (Koç, 2010).

4.3 Bees Algorithm Applications

The Bees Algorithm as described above is applicable to both combinatorial and functional optimisation problems so the performance of the Bees Algorithm was tested on continuous and combinatorial problems.

4.3.1 Continuous domains applications

Several continuous applications of the Bees Algorithm are including functional optimization problems with mathematical test functions are given below:

- De Jong's function, Shekel's Foxholes and Schwefel's function. (Koç, 2010). These problems were used to test the Bees Algorithm and establish the correct values of its parameters and seven problems for benchmarking the algorithm.
- Mathematical benchmarks functions (Ghanbarzadeh 2007; Koç 2010; Sholedolu 2009) and eight benchmark functions (Mathur, 2000).

  The results compared with those obtained using other optimisation algorithms. The test functions and their optima are shown in below:

Table 4.2 : Test Functions (Mathur, 2000).

No	Function Name	Interval	Function	Global Optimum
1	De Jong	[-2.048, 2.048]	$\max F = (3905.93) - 100(x_1^2 - x_2^2) - (1 - x_1)^2$	X(1,1) F=3905.93
2	Goldstein & Price	[-2, 2]	$\min F = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $X[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	X(0,-1) F=3
3	Branin	[-5, 10]	$\min F = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1-f)\cos(x_1) + e$ $a=1, b=\frac{5.1}{4}\left(\frac{7}{22}\right)^2, c=\frac{5}{22}X7, d=6, e=10, f=\frac{1}{8}X\frac{7}{2}$	X(-22/7,12.275) X(22/7,2.275) X(66/7,2.475) F=0.3977272
4	Martin & Gaddy	[0, 10]	$\min F = (x_1 - x_2)^2 + ((x_1 + x_2 - 10)/3)^2$	X(5,5) F=0
5	Rosenbrock	[-1.2, 1.2] [-10, 10]	$\min \ F = 100\ (x_1^2 - x_2)^2 + (1 - x_1)^2$	X(1,1) F=0
6	Rosenbrock	[-1.2, 1.2]	$\min \ F = \sum_{i=1}^{3} \{100\ (x_i^2 - x_{i+1})^2 + (1 - x_i)^2\}$	X(1,1,1,1) F=0
7	Hyper sphere	[-5.12, 5.12]	$\min F = \sum_{i=1}^{6} x_i^2$	X(0,0,0,0,0,0) F=0

Table 4.3 : Results (Mathur, 2000).

	SIMPSA		NE SIMPSA		GA		ANTS		The Bees Algorithm	
func no	success %	mean no. of eval.	success %	mean no. of eval.	success %	mean no. of eval.	success %	mean no. of eval.	success %	mean no. of eval.
1	***	***	***	***	100	10160	100	6000	100	1210
2	***	***	***	***	100	5662	100	5330	100	999
3	***	***	***	***	100	7325	100	1936	100	1657
4	***	***	***	***	100	2844	100	1688	100	526
5a	100	10780	100	4508	100	10212	100	6842	100	898
5b	100	12500	100	5007	***	***	100	7505	100	2306
6	99	21177	94	3053	***	***	100	8471	100	29185
7	***	***	***	***	100	15468	100	22050	100	7113

*** Data not available

Table 4.2 presents the results obtained by the Bees Algorithm and those by the deterministic Simplex method (SIMPSA), the stochastic simulated annealing optimisation procedure (NE SIMPSA, the Genetic Algorithm (GA) and the Ant Colony System (ANTS) (Mathur, 2000). Again, the numbers of points visited shown are averages for 100 independent runs.

- Neural network training for a variety of industrial applications and recursive filter design.
- Mechanical design like desing of welded beam, desing of coil spring (Ang 2009; Pham and Ghanbarzadeh 2007),
- Wood defect classification (Pham and Haj Darwish 2010; Pham, 2007c; Pham 2006b).
- Environmental/Economic Power Dispatch Problems (EEDP) (Lee and Haj Darwish 2008)
- Chemical engineering process (Pham et al., 2008)
- Digital Filter Optimization
- Function Optimizastion

4.3.2 Combinatorial domains applications

There are lots of applications of Bees Algorithm to a combinatorial optimisation problem in the literature is given below below:

- Job Shop Scheduling Problem (JSSP) (Pham, 2007b)
- Wood defect classification (Pham and Haj Darwish 2010; Pham et al., 2007c; Pham et al., 2006b).
- Printed Circuit Board (PCB) problem (Ang, 2010)

Before we focused on details of the Bees Algorithm for combinatorial domains we will give, a important no free lunch theorem that is about mathematical analysis of computing, computational complexity and optimization problems.

4.4 No Free Lunch Theorem

Theorem 4.5.1 (Weak NFL) Given search algorithms A, $A'$and function f $\in Y^X$, there exists a function $f' \in Y^X$ such that $\pi_{[X]}\left(\mathrm{A(f)}_y\right) = \pi_{[X]}\left(A'(f')_y\right)$, that $\pi_{[X]}$ is cardinality.

Definition: *A performance measure* with respect to a set $f \in Y_X$ is any function μ_f defined over the collection of all search algorithms such that $\mu_f(A)$ is a function of the multiset $\{\{A(f)_y : f \in F\}\}$. Search algorithms perform equally well on F if they are evaluated identically by every performance measure with respect to F.

Theorem 4.5.2 (NFL) Every efficient search algorithm performs equally well on F if and only if F is closed.

No-free-lunch theorems may be of theoretical importance, and they can also have important implications for algorithm development in practice. The theorem says that, the fact there is no universally efficient algorithm so if algorithm A performed better than algorithm B in some class of problems, then algorithm B performed better than algorithm A in some other class of problems. On average, each algorithm produced similar performance in respect to other algorithms. In addition, the performance of an algorithm on a set of benchmarking problems did not guarantee giving similar performance on a different class of problems (Wolbert 1997).

Obviously, in reality, the algorithms with problem-specific knowledge typically work better than random search, and that there is no universally generic tool that works best for all the problems. Therefore, we have to seek balance between speciality and generality, between algorithm simplicity and problem complexity, and between problem-specific knowledge of optimization problems.

5. BEES ALGORITHM FOR COMBINATORIAL SPACES

Combinatorial optimization problems have attracted much attention of researchers over the years can generally be defined as problems that require searching for the best solution among a large number of finite discrete candidate solutions. Approximation algorithms, like population-based algorithms are techniques that solve ‘NP-hard’ CO problems in a reasonable amount of computation time.

In this chapter, the Bees Algorithm is presented for combinatorial domains and it was tested on Travelling Salesman Problem with different neighbourhood strategies.

In the basic version of the Bees Algorithm, a kind of neighbourhood search combined with a random search to enable it to locate the global optimum. In combinatorial domains, unlike continuous domains, there is no mathematical distance definition for neighbourhood search. So we use similar but not same version of the Bees Algorithm for continuous domains as we presented in the previous chapter.

In combinatorial domains, the patch idea of the Bees Algorithm for continuos domians replaced by a local search operator to be able to perform a local search the main difference of combinatorial domains. Removing the shrinking procedure is also another difference. However, the abondonment procedure can be used in both of the solution spaces to improve the global search part. The pseudo-code of the Bees Algorithm for combinatorial domains is given in Figure5.1.

1. Initial population with n random solution; random(Sequence(n)).

2. Evaluate fitness of the population.

3. While (stopping criterion not met)

4. Select sites (m) for neighbourhood search.

5. Recruit bees for selected sites (more bees for best e sites), evaluate fitnesses, select the fittest bee from each site and shrink patches

for (k=l ; k=e ; k++) // Elite Sites

for (i=l ; i= nep ; i++) // More Bees for Elite Sites

RecruitedBee(k)(i) = NeighbourhoodOperator(Sequence(k));

Evaluate Fitness = RecruitedBee(k)(i); //Evalute the fitnees of recruited Bee(i)

If (Bee(i) is better than Bee(i-l)) RepresentativeBee = RecruitedBee(k)(i);

for (k=e ; k=m ; k++) // Other selected sites (m-e)

for (Bee=l ; Bee= nsp ; Bee++) // Less Bees for Other Selected Sites (m-e)

RecruitedBee(k)(i) = NghOperator(Sequence(k));

Evaluate Fitness = RecruitedBee(k)(i); //Evalute the fitnees of recruited Bee(i)

If (Bee(i) is better than Bee(i-l)) RepresentativeBee = RecruitedBee(k)(i);

6. If (Iteration > sat)

If (no improvement on the site)

Save the Best Fitness;

Abandon the Site;

Bee(m) = GenerateRandomValue(All Search Space);

7. Assign remaining bees to search randomly and evaluate their fitnesses. // (n-m) assigned to search randomly into whole solution space

8. End While

Figure 5.1 : The pseudo-code of the Bees Algorithm for combinatorial domains (Koç, 2010).

5.1 The Travelling Salesman Problem

Traveling Salesman Problem (TSP) is about finding a Hamiltonian path with minimum cost. This cost is referred to as the tour length. Since it is the length of the tour a salesman would make when visiting the cities in the order specified by the permutation, returning at the end to the initial city.

Definition 5.1 A graph G is a composite of a set V_G of vertices and another set E_G of edges, where an edge is a set of two distinct vertices. For example we may have V_G = {1, 2, 3, 4} with E_G = {{1, 2}, {1, 3}, {2, 3}, {2, 4}}.

Definition 5.2 Let G = (V,E) be an undirected graph. A Hamiltonian cycle of G is a cycle that visits every vertex v ∈ V exactly once. Instead of Hamiltonian cycle, we sometimes also use the term tour.

Suppose a salesman is given a set of cities associated with traveling distances (or costs) from any city to any other city.

The salesman must visit every city only once and then return to the starting city with minimum distances (or costs). Given a starting city, it has $(V-1)$ choices for the second city, $(V-2)$ choices for the third city, etc. Multiplying these together one gets $(V-1)!$ for one city and $(V)!$ for the V cities. Another solution is to try all the permutations (ordered combinations) and see which one is cheapest. At the end, the order is also factorial of the number of cities. Briefly, the solutions which appear in the literature are quite similar. The TSP is therefore to determine a Hamiltonian tour with minimum cost that is one of the discrete optimization problems which is classified as NP-hard combinatorial optimization problem.

Definition 5.3 Let G = (V,E) be a graph. V is a set of m cities, V = $\{v_1, \ldots, v_m\}$ and E is a set of arcs or edges, $E = \{(i,j): i,j \in V\}$.

Remember that to formulate an optimization model, we need to define a search space and search space contains the set of feasible solutions of an optimization problem. Furthermore, a search space can define relationships (for example distances) between solutions.

Very generally, a search space can be defined as a *topological space.*A topological space is a generalization of metric search spaces (as well as other types of search spaces)

Definition 5.4 A topological space is an ordered pair (X, T), where X is a set of solutions (points) and T is a collection of subsets of X called open sets. A set Y is in X (denoted $Y \subseteq X$) if every element $x \in$ Y is also in X (x ∈ Y ⇒ x ∈ X).

A topological space (X, T) has the following properties:

1. the empty set Ø and whole space X are in T,

2. the intersection of two elements of T is again in T, and

3. the union of an arbitrary number of elements of T is again in T.

Metric search spaces are a specialized form of topological spaces where the similarities between solutions are measured by a distance. Therefore, in metric search spaces, we have a set X of solutions and a real-valued distance function also called a *metric*: $d : X \times X \rightarrow \mathbb{R}$ that assigns a real-valued distance to any combination of two elements $x, y \in X$. In metric search spaces, the following properties must hold:

1. $d(x, y) \geq 0$,

2. $d(x, x) = 0$,

3. $d(x, y) = d(y, x)$,

4. $d(x, z) \leq d(x, y) + d(y, z)$ $(triangle\ inequality)$ $where\ x, y, z \in X$.

Example 5.1 example of a metric that can be defined on $\mathbb{R}^n$ is the Euclidean metric. In Euclidean spaces, a solution x = $(x_1, \ldots, x_n$ is a vector of continuous values ($x_i \in \mathbb{R}$). The Euclidean distance between two solutions x and y is defined

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad \textbf{(5.1)}$$

For n = 2, we have a standard 2-dimensional search space and the distance between two elements $x, y \in \mathbb{R}^2$. Many optimization models use metric search spaces. A metric search space is a topological space where a metric between the elements of the set X is defined. Therefore, we can define similarities between solutions based on the distance d.

Definition 5.5 (Metric TSP) Let G be a complete undirected graph G with a weights $d : E(G) \rightarrow \mathbb{R}_+$ that satisfy the triangle inequality $d(u,w) \leq d(u,v) + d(v,w)$ for all $u, v, w \in V(G)$. E is normally associated with a distance (or cost) matrix, which is defined as If $d_{i,j} = d_{j,i}$ the problem is a symmetric TSP (STSP). Otherwise, it becomes an asymmetric TSP (ATSP).

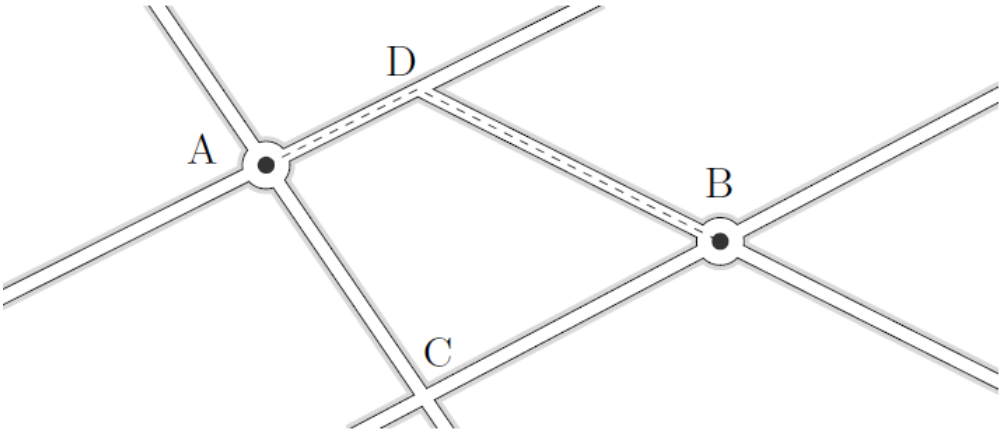
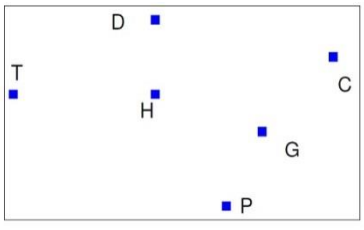


Figure 5.2 : Triangularity in a road network. The distance from A to B is determined by the shortest route $d(A,B) \leq d(A,X) + d(X,B)$ for every X (Hetland, 2009).

Example 5.2: Sabrina has the following list of errands:

- Pet store (the black cat needs a new litterbox) (P)
- Greenhouse (replenish supply of deadly nightshade) (G)
- Pick up black dress from cleaners (C)
- Drugstore (eye of newt, wing of bat, toothpaste) (D)
- Target (weekly special on cauldrons) (T)



In witch which order should Sabrina do these errands in order to minimize the time spent?

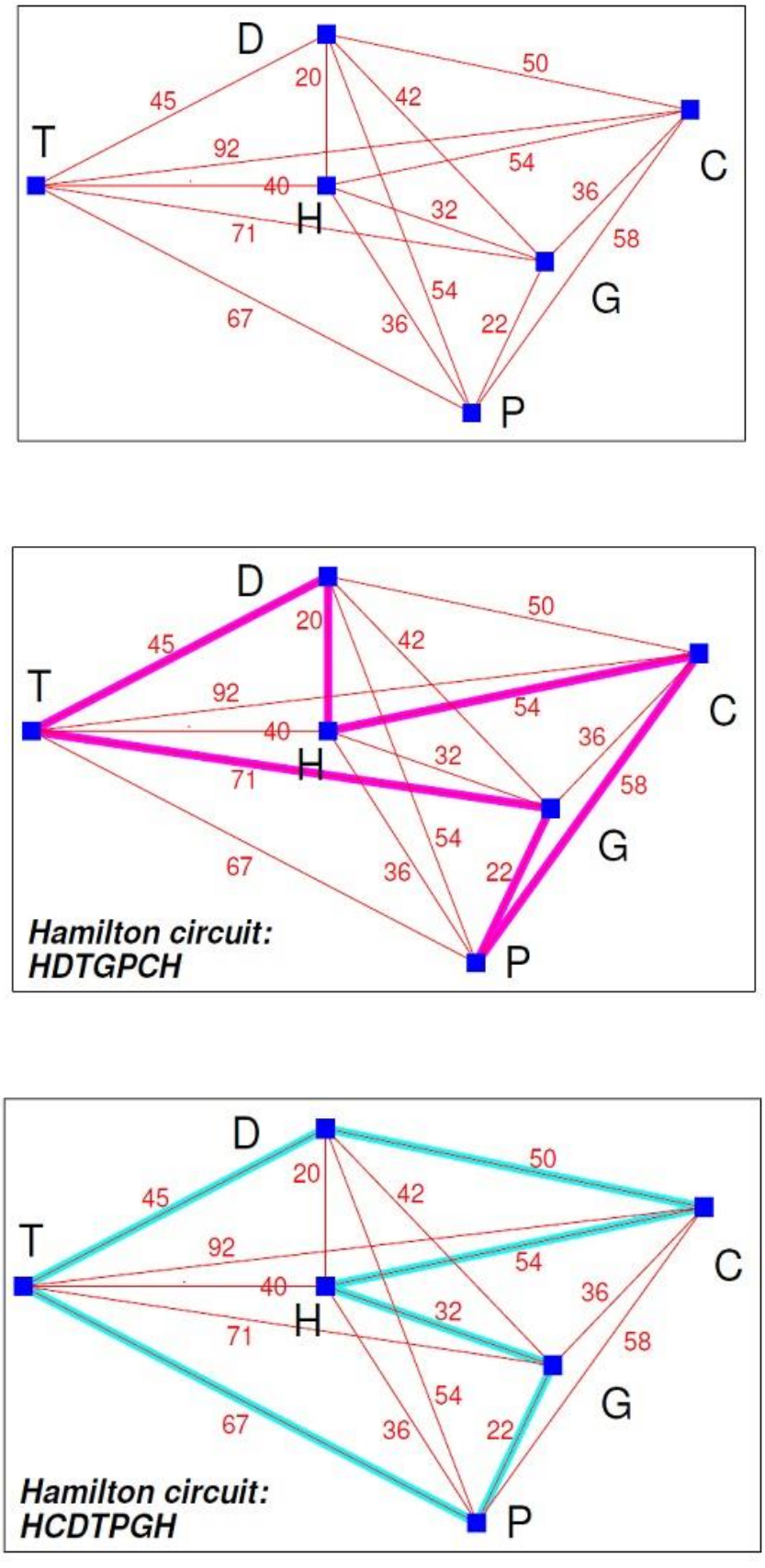


Figure 5.3 : The Hamiltonian path minimize the time spent.

5.2 Proximity Queries in Metric Spaces

Let D be a set, d a distance function defined on D, and $M = (D, d)$ a metric space. Given a set $X \subseteq D$, structure the elements of X so that similarity queries can be answered basically three types of queries in metric spaces:

1. Find objects whose feature values fall within a given range or where the distance, using a suitably defined distance metric, from some query object falls into a certain range (range queries).
2. Find objects whose features have values similar to those of a given query object or set of query objects (nearest neighbor queries). In order to reduce the complexity of the search process, the precision of the required similarity can be an approximation (approximate nearest neighbor queries).
3. Find pairs of objects from the same set or different sets which are sufficiently similar to each other (closest pairs queries).

5.2.1 Range query

This is the most common type of query that is meaningful almost in every application. The query $R(q, r)$ is specified by the query object q and the query radius r and retrieves all objects which are within the distance of r from q, shown in Figure5.4:

$$R(q, r) = \{u \in \mathbb{U} | d(q, u) \leq r\} \quad \textbf{(5.2)}$$

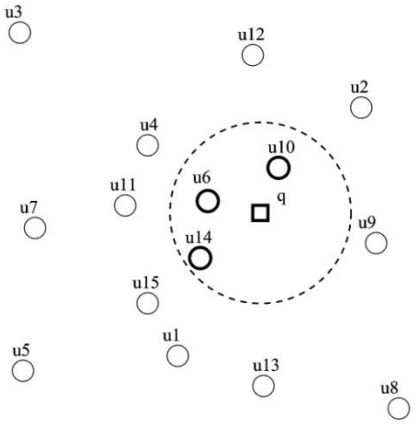


Figure 5.4 : $R(q, r)$ retrieves all objects which are within the distance of r to the query object q (Chávez, 2001).

5.2.2 Nearest neigbor query (NN(q))

This query finds one nearest neighbor, that is, the object closest to the given query object. In general case where for k nearest neighbors that is, k-NN(q) query retrieves k nearest neighbors to the object q:

$$NN(q) = \{u \in \mathbb{U} | \forall\, v \in \mathbb{U}\,, d(q,u) \leq d(q,v)\}$$

$$k - NN(q) = \{R \subseteq X; |R| = k \; and \; \forall x \in R;\, y \in X - R : d(q;\, x) \leq d(q;\, y)\}$$

In case of $k - NN(q)$ we are satisfied with any set of k elements satisfying the condition. Here we select the q as a *pivot element* sometimes called centers.

5.3 Neighbourhood Strategies

Definition 5.4 A neighbourhood is a function

$$N: S \rightarrow 2^S$$

that assigns to every $s \in S \; a \; set \; of \; neighbors \; N(s) \subseteq S$ and N(s) called the neigbourhood of s, S is the search space containing all possible solutions.

A neighborhood definition can be viewed as a mapping that assigns to each solution s¸ S a set of solutions y that are neighbors of s.

There are several exchange neighbourhood strategies and local search algorithms in the literature. Among simple local search algorithms, the most famous are 2-Opt and 3-Opt and insert ect. that swap operators are considered as exchange neighbourhood strategies (Aarts and Lenstra, 1997). They simply change the position of a randomly selected city to create an altered path. By contrast, 2-Opt and 3-Opt are simple local search algorithms that delete two or three edges, thus breaking the tour into two paths and then reconnecting those paths later.

These approaches can be roughly divided into local (heuristic) search and global search approaches. Some of the local search approaches are such as 2-opt, 3-opt (Lin, 1965). The global search approaches, such as simulated annealing (KirkkPatrick, 1983), Hopfield neural networks, and evolutionary algorithms (Nagata, 1997) , (Freisleben, 1996), (Dorigo, 1997), (Tao, 1998), (Mulhem, 1998) ,(Zhenya, 1999) have been proposed to reduce the ill effect of these local search methods, but they often converge more slowly compare to local search approaches (Tsai, 2002).

2-Opt algorithm deletes two edges, thus breaking the tour into two paths, and then reconnects those paths in the other possible way. See Figure 5.2.

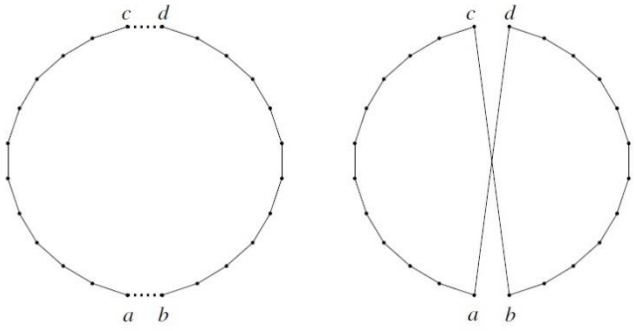


Figure 5.5 : A 2-Opt move: original tour on the left and resulting tour on the right (Johnson, 1997).

3-Opt algorithm deletes three edges, thus breaking the tour into three paths, and then reconnects those paths in the other possible way. See Figure 5.4.

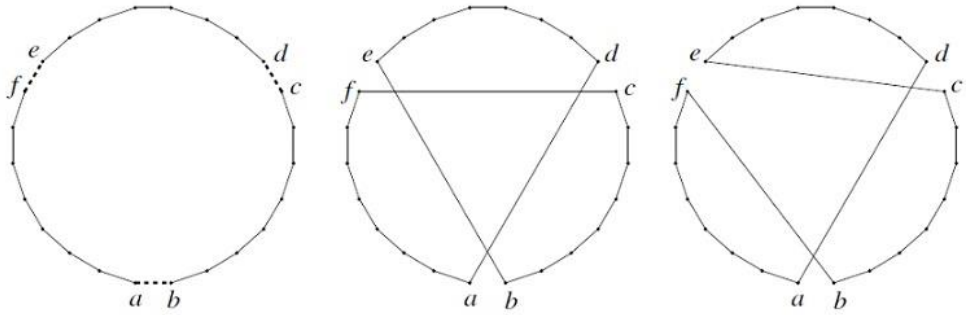


Figure 5.6 : A 3-Opt move: original tour and resulting tour (Johnson, 1997).

We don't necessarilly have to stop at 3-opt, we can continue with 4-opt and so on, but each of these will take more and more time and will only yield a small improvement on the 2- and 3-opt operator.

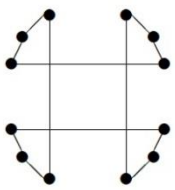


Figure 5.7 : The double bridge move 4-opt move is called "the crossing bridges" (Davendra, 2010).

6. VANTAGE POINT NEIGHBOURHOOD SEARCH IN THE BA

6.1 Preliminaries

The Nearest Neighbor field includes the study of decision making and learning based on neighborhoods, the underlying metrics and spaces, and the matter of computing/searching for the neighborhood about a point (Yianilos,1993). Searching includes several forms of vantage point tree (vp-tree). It is the data structure introduced in several forms, together with associated algorithms, as an improved method for these difficult search problems.

Each element of metric space distances to every other element formed a perspective on the entire space. *Vantage points* sometimes called pivot element cuts/ divide the entire space and it formed a vantage point tree.

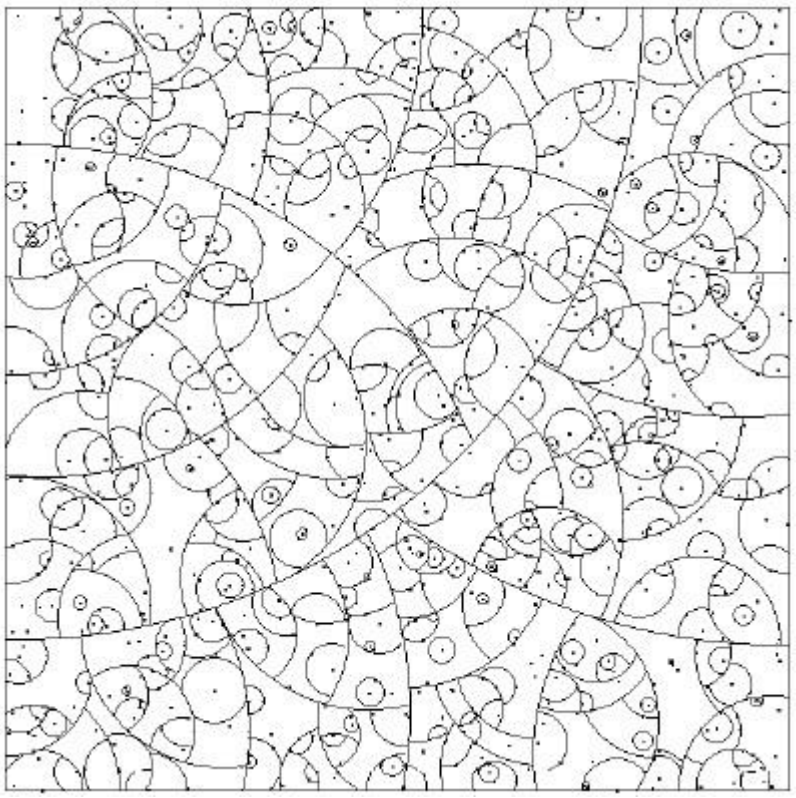


Figure 6.1 : Vantage point decomposition (Yialinos, 1993).

Vantage Point Trees (vp-tree) formed by simplest algorithm. Its distinguished vantage point then splits the space into left and right space. This is building a binary tree recursively, taking *any element p* as the root (vantage point) and taking the median of the set of all distances, M =$median\{ d(u,p)|u \in \mathbb{U}\}$. The left space or left subtree contains the elements u, which satisfied $d(u,p) \leq M$ and right subtree contains the elements u, which satisfied $d(u,p) > M$. The algorithm construct this subtrees with selecting a vantage point element ramdomly.

The VPT takes *O(n)* space and is built in *O*(*n*log*n*) worst case time, since it is balanced. We first measure $d = d(q,p)$. If $d - r \leq M$ enter the element to the left space, if $d + r > M$ enter it to the right. Notice that this selecting algorithm can be used both for discrete distance functions and continuous distance functions.

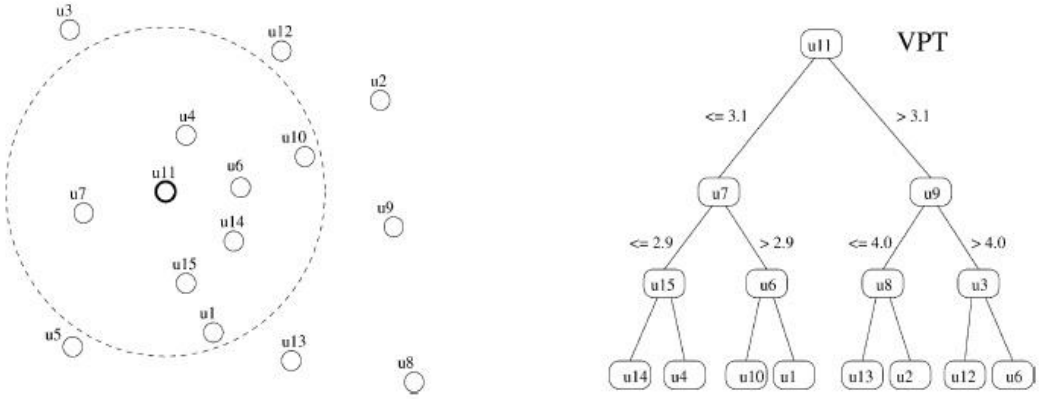


Figure 6.2 : Example VPT with root u_{11} (Chávez, 2001).

6.2 The Bees Algorithm with Vantage Point Neihbourhood Search

The simplest vp-tree construction begins with selecting a pivot element, vantage point randomly. Given a set S of metric space elements (i.e combinatorial search space elements), the algorithm returns pointer to the root of an optimized vp-tree that satisfied the local optimum value (for example in TSP returns the optimal tour for each iteration of recruit phase).

The algorithm of making vantage point neighbourhood searh for the Bees Algorithm recruitment phase is given below:

1. Recurse the following steps until all sites are chosen.

2. Select a vantage point p (pivot site) randomly from the all sites.

Add vantage point to the solution list; // i.e hist list

3. Calculate median of the set of all distances, M = $median\{ d(s,p) | s \in S\}$.

// return a list hist of the distances from the item to each vantage point

4. Splits site list into two list L and R. Take only left site list (L) for optimal solution.

for (site=1; site < allSites; site ++)

if ($d(site, p) <$ median) add site to L,

5. Select new vantage point from the L randomly. Add pivot site to the solution list and delete selected site from all sites.

6. Go to step 3, and repeat until convergence or termination conditions are met.

7. Return solution site list for evaluating fitness.

Figure 6.3 : Proposed pseudo-code of the Vantage Point Bees Algorithm recruitment phase.

This algorithm presents a modification of the neighbourhood search procedure in the Bess Algorithm for combinatorial domains. We proposed vantage point neighbourhood search procedure for the Bees Algorithm local search in the recruitment selection.

Like the original Bess Algorithm, this new algorithm required the same six parameters (n, m, e, nsp, nep, ngh). Initially, a number of bees (n) were sent randomly to the search space. Each bee was associated with one solution. The solutions representing the fitness of individual bees were then ranked in descending order. The top m solutions were regarded as selected sites. Of m sites, a number of top e site(s) were considered as elite one(s). Each of non-elite (m-e) and elite (e) sites respectively received nsp and nep forager bee(s) to exploit the discovered food source. All this steps are the same with the original Bees Algorithm.

To develop a new local search in the recruitment phase we use the vantage point tree algorithm with median calculations. The Bees Algorithm with vantage point neihbourhood procedure is suggested as an addition to the Bees Algorithm to deal

with combinatorial domains. The algorithm is applied to the Travelling Salesman Problem (TSP) to show that the algorithm is both robust and efficient.

1. Initial population with n random solution.

2. Evaluate fitness of the population.

3. While (stopping criterion not met)

4. Select sites (m) for neighbourhood search.

5. Recruit bees for selected sites (more bees for best e sites), evaluate fitnesses, select the fittest bee from each site and shrink patches

```
for (k=l ; k=e ; k++) // Elite Sites
  for (Bee=l ; Bee= nep ; Bee++) // More Bees for Elite Sites
BeesPositionlnNghO = GenerateVantagePointTree (Bee(i), all sites),
Evaluate Fitness = Bee(i); //Evalute the fitnees of recruited Bee(i)
If (Bee(i) is better then Bee(i-l)) RepresentativeBee = Bee(i);
for (k=e ; k=m ; k++) // Other selected sites (m-e)
  for (Bee=l ; Bee= nsp ; Bee++) // Less Bees for Other Selected Sites (m-e)
BeesPositionlnNghO = GenerateVantagePointSolution (Bee(i) ,all sites),
        Evaluate Fitness = Bee(i); //Evalute the fitnees o f recruited Bee(i)
        If (Bee(i) is better then Bee(i-l))
RepresentativeBee = Bee(i);
```

6. If (Iteration > sat)

```
If (no improvement on the site)
    Save the Best Fitness;
    Abandon the Site;
     Bee(m) = GenerateRandomValue(All Search Space);
```

7. Assign remaining bees to search randomly and evaluate their fitnesses.

// (n-m) assigned to search randomly into whole solution space

8. End While

Figure 6.4 : Proposed Pseudo-code of the Vantage Point Bees Algorithm.

6.3 A Proposed VPBA for TSP and Experimental Results

The performance of the VPBA is investigated by appliying the algorithm to benchmark problem taken from TSPLIB. As an instance we choose Eil51, that is a 51-city TSP problem and we compare the test result with the performance of the Bees Algorithm with several local search operators including simple (2 point) swap, double (4 point) swap, insert, 3 point swap, 2-Opt and 3-Opt.

The experiments were performed using the Vantage Point Bees Algorithm to evolve its own parameter values. It was run 50 times for each parameter setting on eil51 benchmark problem.

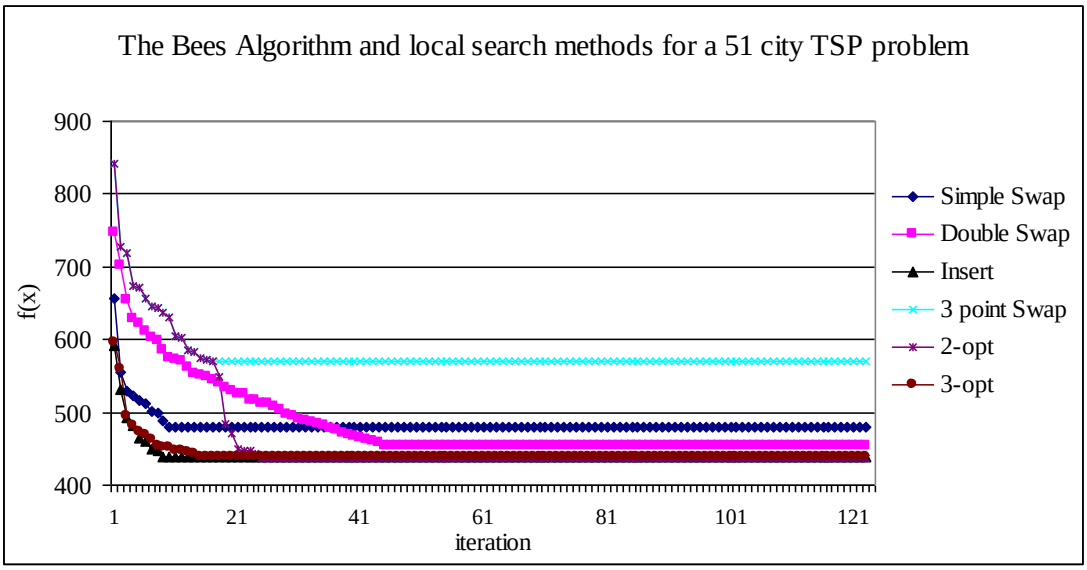
The computing platform used to perform the experiments was a 2.50GHz Intel(R) Core(TM) i5-2450M CPU PC with 4 GB of RAM. The experimental programs were coded in the Java language and compiled with Eclipse IDE. Each problem instance was run across 50 random seeds. The parameters of Vantage Point Bees Algorithm for eil51 TSP shown in Table 6.1:

Table 6.1 : Parameters for VPBA of eil51 TSP.

Parameter	**Symbols**
Number of scout bees in the selected patches	n = 80
Number of best patches in the selected patches	m = 20
Number of elite patches in the selected best patches	e = 5
Number of recruited bees in the elite patches	nep = 80
Number of recruited bees in the non-elite best	nsp = 40
Number of iterations	50

Table 6.2 summarizes the results of the Bees Algorithm with (2 point) swap, double (4 point) swap, insert, 3 point swap, 2-Opt and 3-Opt operator on eil51 TSP. (Koç,2010).

Table 6.2 : Performance of the Bees Algorithm with different local search methods.



The eil51 problem was tested 50 independent runs. From the experimental result we select the best tour length, average tour length, and calculate standard deviation of trails are used to measure the performance of comparative methods. where solution is the experimental value and optimum is the optimum of a TSP problem. For each problem the proposed algorithm can find the best tour in almost each trial and the error rate is only 0.02% away from the optimal.

The average Δ_{avg} for the 50 runs was computed as follows:

$$\Delta_{avg} = \left(\sum_{i=1}^{R} \frac{(F_{BA} - F_{ref})}{F_{ref}} \times 100 \right) / R \qquad \textbf{(6.1)}$$

Where $F_{BA}$is the fitnees value generated by the VPBA in each run , F_{ref} is te reference fitness value from eil51 TSP in TSPLIB and R=50 is the total number of runs,respectively. Δ_{std} denotes the standard deviation over the R runs.

Table 6.3 : Benchmark results for 51 city TSP with VPBA.

R=number of runs	Δ_{avg}	Δ_{std}
50	2,42723	1,14358

To investigate the performance of vantage point Bees Algorithm (VPBA) on these

50 independent runs, we compare the for the best case tour (best_tour) and average case tour (average_tour) and the worst case tour (worst_tour) is chosen see the following graph:

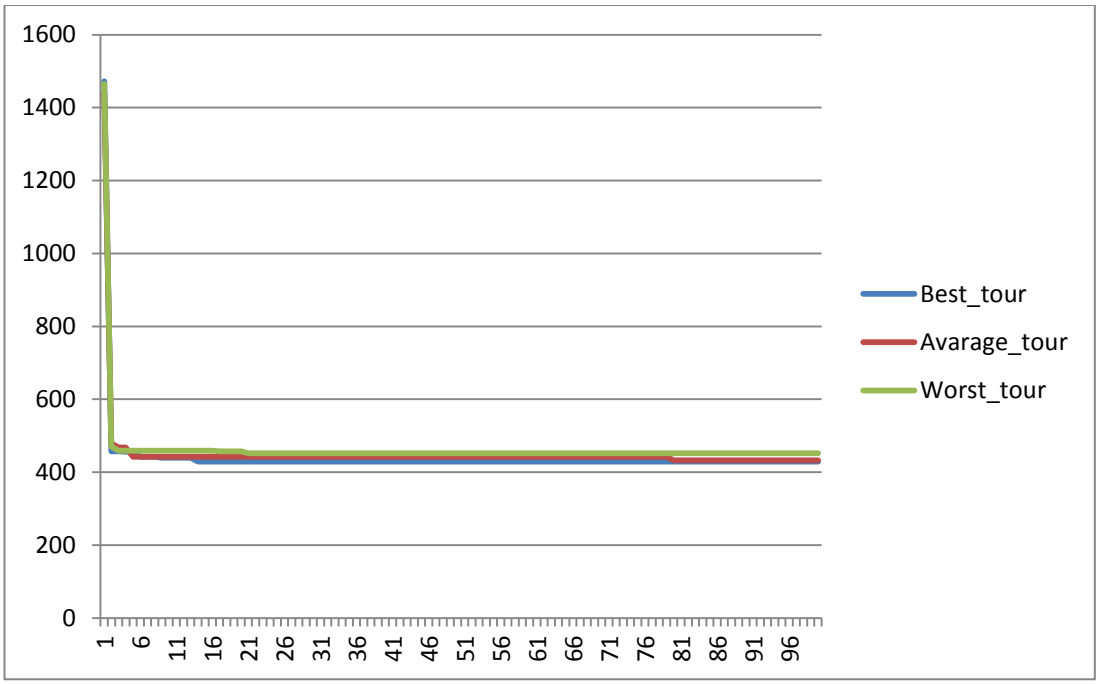


Figure 6.5 : Performance of the Vantage Point Bees Algorithm.

6.4 Conclusions

The performance of the VPBA was significantly fast in finding the optimal optimum of tested benchmark function.

The performance of the Vantage Point Bees Algorithm was evaluated using 51-city TSP and the results were compared to original Bees Algorithm with several local search operators including simple (2 point) swap, double (4 point) swap, insert, 3 point swap, 2-Opt and 3.

The results can be improved and the VPBA performs well against one that uses standard/fixed parameter values. This is attributed to the fact that parameter values suitable for a particular problem instance can be automatically derived and varied throughout the search process.

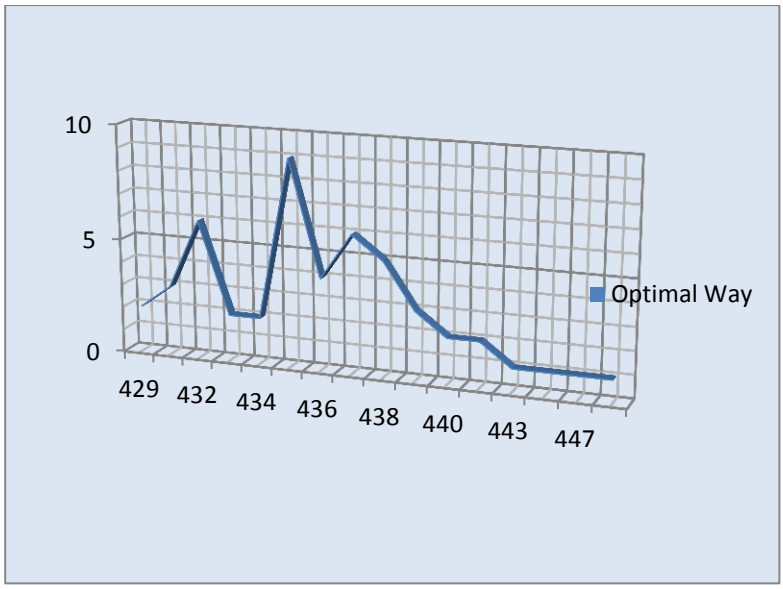


Figure 6.6 : Distribution graph of eil51 TSP problem with the VPBA.

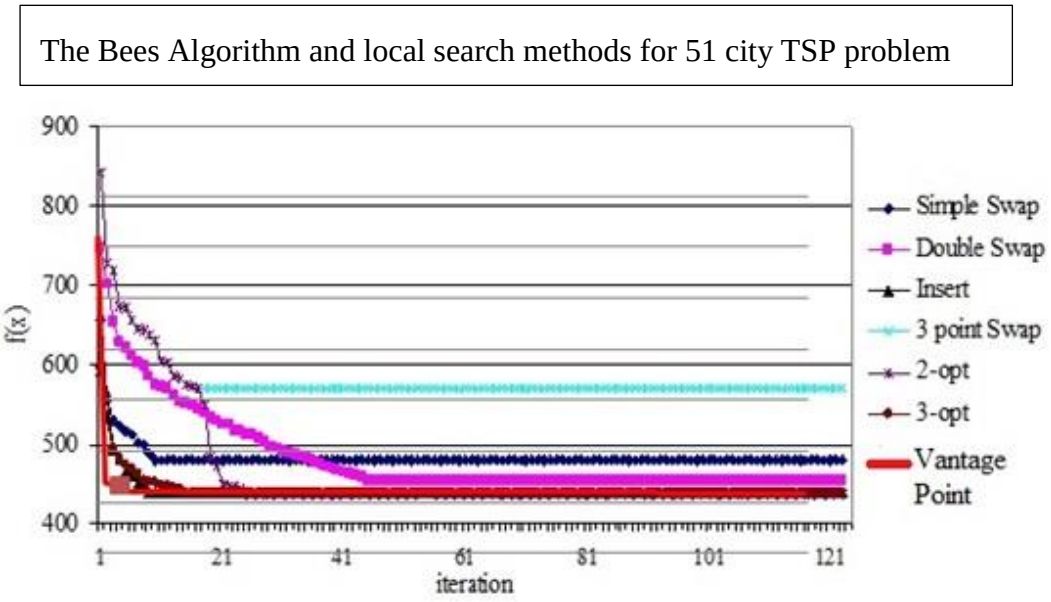


Figure 6.7 : This graph compare the performance of the BA with several local search operators including simple (2 point) swap, double (4 point) swap, insert, 3 point swap, 2-Opt and 3-Opt (Koç, 2010) with the performance of Vantage Point .

REFERENCES

Aarts, E. and Lenstra, J. K. (1997). Local search in combinatorial optimization Joh Wiley & Sons Ltd, England.

Alizadeh, F., Karp, R.M., Newberg, L.A., and Weisser, D.K., (1993). Physical mapping of chromosomes: a combinatorial problem in molecular biology." In Proc. 4th ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 371-381.

Blackwell, T. and Branke, J., (2004). Multi-swarm optimization in dynamic environments, Applications of Evolutionary Computing, in Lecture Notes in Computer Science, Raid,l G.R., Editor, Springer-Verlag: Berlin, Germany.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). Swarm intelligence: from natural to artificial systems. New York: Oxford University Press.

Bozkaya T. and Ozsoyoglu. M. (1997) Distance-based Indexing for High-Dimensional Metric Spaces. In ACM SIGMOD Record, volume 26, pp. 357–368,

Camazine, S. and Sneyd, J. (1991). A model of collective nectar source selection by honey bees: self-organization through simple rules. Journal of Theoretical Biology, pp. 547-571.

Camazine, S., Deneubourg, **J., Franks, N. R., Sneyd, J., Theraula, G. And Bonabeau, E**. (2003). Self-organization in biological systems. Princeton: Princeton University Press.

Chávez, E. E., Marroqu´ın, M., J, and Yates. R., B., (1999) An array based algorithm for similarity queries in metric spaces. In Proceedings of the String Processing and Information Retrieval Symposium &International Workshop on Groupware (SPIRE), pages 38–46. IEEE Computer Society.

Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. L. (2001). Searching in metric spaces. ACM computing surveys (CSUR), 33(3), pp. 273-321.

Eberhart, R., Shi, Y. and Kennedy, J. (2001). Swarm intelligence. San Francisco: Morgan Kaufmann Publishers.

Engelbrecht, A. P. (2005). Fundamentals of computational swarm intelligence. Hoboken, N.J. Wiley.

David S J. and Lyle A Mcgeoch, The Traveling Salesman Problem : A Case Study in Local Optimization, pp. 1–103.

Davendra, D., Zelinka, I., Senkerik, R., & Bialic-Davendra, M. (2010). Chaos driven evolutionary algorithm for the traveling salesman problem. Traveling salesman problem, theory and applications. InTech Publishing, Croatia, pp. 55-70.

De Castro, L., N., D., Zuben, F., J., V., (1999). Artificial Immune Systems, Technical Report.

De Castro, L. N., and Timmis, J. (2002). Artificial immune systems: a new computational intelligence approach. Springer.

Dorigo, M., Maniezzo, V. and Colomi, A. (1991). Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Dorigo, M., Maniezzo V. and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B 26(1): pp. 29-41.

Dorigo, M., and Gambardella, L. M., (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE. Transaction on Evolutionary Computation, vol.1, no.1, pp. 53-66.

Dorigo, M., and Di Caro, G. (1999). The ant colony optimization metaheuristic. New Ideas in Optimization, pp. 11-32.

Dorigo, M., Di Caro, G., Gambardella, L.M., (1999), Ant Algorithms for Discrete Optimization, Journal of Artificial Intelligence Research, 5(2), 137-172.

Dorigo, M. and Stützle, T., (2004). Ant colony optimization, Cambridge: MIT Press.

Fogel, L.J., Owens, A.J., and Walsh, M.J. (1966). Artificial intelligence through simulated evolution, J. Wiley, New York.

Fogel, D.B. (2000). Evolutionary computation: toward a new philosophy of machine intelligence, 2nd edition, New York: IEEE Press.

Freisleben, B. and Merz, P., (1996). New genetic local search operators for the traveling salesman problem. In Proc. PPSN IV- 4th Int. Conf Parallel Problem Solving from Nature. Berlin, Germany: Springer-Verlag, pp. 890-899.

Frisch, K. (1967). The Dance Language and Orientation of Bees. Cambridge, Mass.: The Belknap Press of Harvard University Press.

Ghanbarzadeh, A. (2007). The Bees algorithm. A novel optimisation tool. PhD. Cardiff University.

Greco, F., (2008) Travelling Salesman Problem, *Traveling Salesman Problem*, ed. by InTech.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison Wesley, MA.

Johnson, D. S., and McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. Local search in combinatorial optimization, pp. 215-310.

Haddad, O.B., Afshar, A. and Marino, M.A., (2006). Honeybees mating optimization (HBMO) algorithm: A new heuristic approach for water resources optimization, Water Resources Management, (20), pp. 661–680.

Hazem A., Janice G. and Canada K., (2012). Swarm Intelligence : Concepts , Models and Applications Technical Report.

Hetland, M. L. (2009). The Basic Principles of Metric Indexing, pp. 22–29.

Holland, J.H. (1975). Adaptation in natural and artificial systems, University of Michigan Press, Ann Arbor.

Hopfield, J. J. and Tank, D. W., (1985). Neural computation of decisions in optimization problems. Biological Cybernetics, vol. 52, pp. 141-152.

Hosny, M., I., (2010). Investigating Heuristic and Meta-Heuristic Algorithms for Solving Pickup and Delivery Problems Manar Ibrahim Hosny School of Computer Science & Informatics.

Hunt, J. E., and Cooke, D. E. (1996). Learning using an artificial immune system. Journal of network and computer applications, 19(2), pp. 189-212.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga, D. and Akay, B., (2007). Artificial bee colony (ABC) algorithm on training artificial neural networks. Proc IEEE 15th Signal Processing and Communications, Applications, pp. 1-4.

Karaboga, D. and Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. Applied Soft Computing 8(1), pp. 687-697.

Karaboga, D. and Akay, B. (2009). A comparative study of artificial bee colony algorithm, Appl. Math. Comput.

Karaboga, D. and Gorkemli B. (2011) "A Combinatorial Artificial Bee Colony Algorithm for Traveling Salesman Problem," 2011 International Symposium on Innovations in Intelligent Systems and Applications, pp. 50–53.

Kennedy, J. and Eberhart, R. (1995) Particle swarm optimization. In Proceedings of the 1995 IEEE International Conference on Neural networks, Perth, Australia, 1995, vol. 4, pp. 1942–1948.

Kennedy, J. and Ebemhart, R. C. (2001). Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco.

Kirkpatrick, S., Gelatt C. D., and Vecchi M. P., (1983), Optimization by simulated annealing." Sci, vol. 200, pp. 671-680.

Koç, E., (2010). The Bees Algorithm Theory , Improvements and Applications, PhD. Manufacturing Engineering Centre School of Engineering University of Wales, Cardiff, United Kingdom.

Korostensky, C. and Gonnet, G. H., (2000). Using traveling salesman problem algorithms for evolutionary tree construction. Bioinformatics Vol. 16 no. 7, pp. 619-627.

Krishnanand, K. N. and Ghose, D., (2009). Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence* 3(2), pp. 87-124.

Lin, S., (1965). Computer solutions of the traveling salesman problem. Bell Syst., J., vol. 23, pp. 2245-2269.

Lee, J. Y. (2010). Multi-Objective Optimisation using the Bees Algorithm. Cardiff University.

Madureira A., Sousa N., Pereira, I. (2005). Swarm Intelligence for Scheduling: a Review, pp. 1–8.

Mathur, M., Karale, S. B., Priye, S., Jayaraman, V. K. and Kulkarni, B. D., (2000). Ant colony approach to continuous function optimization. Ind. Eng. Chem. Res. 39(10), pp. 3814-3822.

Mulhem, M., and Maghrabi, T., (1998). Efficient convex-elastic net algorithm to solve the Euclidean traveling salesman problem. Systems, Man and Cybernetics, Part B, IEEE Transactions, Vol.284, pp. 618 –620.

Xiong, N., Yang, J. He, Y. , Kim, Y. He, T. and Lin, C. (2010) A Survey on Decentralized Flocking Schemes for a Set of Autonomous Mobile Robots (Invited Paper). Journal of Communications, Vol. 5, No. 1, pp. 31-38.

Passino, K., M., (2002). Biomimicry of bacterial foraging for distrubuted optimization and control.Dept. of Electr. Eng., Ohio State Univ., Columbus, OH, USA.

Passino, K.M., Seeley, T.D., and Visscher, P.K., (2008). Swarm cognition in honeybees, Behavioral Ecology Sociobiology, 62(3), pp. 401-414.

Panigrahi, B., K., Shi, Y., Lim, M. Handbook of Swarm Intelligence*:* Concepts,Principles and Applications. Springer.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S. and Zaidi, M. (2005). The Bees Algorithm Technical Report, Cardiff, Manufacturing Engineering Centre, Cardiff University.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S. and Zaidi, M. (2006a).The Bees Algorithm - A Novel Tool for Complex Optimisation Problems.

Pham, D. T. and Ghanbarzadeh, A. (2007), Multi-Objective Optimisation using the Bees Algorithm. Proceedings 3rd International virtual conference on Innovative Production Machines and Systems (IPROMS). 2-13 July 2007. pp. 529-533.

Pham, D. T., Afify, A. and Koç, E. (2007a). Manufacturing Cell Formation using the Bees Algorithm. Proceedings 3rd International Virtual Conference on Innovative Production Machines and Systems (IPROMS 2007), Scotland, pp. 523-528.

Pham, D. T., Koç, E., Lee, J. Y., and Phrueksanant, J. (2007b). Using the bees algorithm to schedule jobs for a machine. In Proceedings of eighth international conference on laser metrology, CMM and machine tool performance, pp. 430-439.

Pham DT, Otri S, Afify A, Mahmuddin M, and Al-Jabbouli H., (2007). Data clustering using the Bees Algorithm, Proc 40th CIRP Int. Manufacturing Systems Seminar, Liverpool.

Pham, D.T. and Sholedolu, M. (2008). Using a hybrid PSO-Bees Algorithm to train Neural Networks for Wood Defect Classification. In: 4th International Virtual Conference on Intelligent Production Machines and Systems, IPROMS.

Pham, D. T. and Castellani, M. (2009). The bees algorithm: Modelling foraging behaviour to solve continuous optimization problems. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 223(12), pp. 2919-2938.

Pham, D. T. and Haj Darwish, A. (2010). Using the bees algorithm with Kalman filtering to train an artificial neural network for pattern classification. Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering 224(7), pp. 885-892.

Otri S., (2011). Improving the Bees Algorithm for Complex Optimization Problems. A Thesis Submitted to the Cardiff University In Candidature for the Degree of Doctor of Philosophy.

Read, M., Andrews, P. S., & Timmis, J. (2012). An Introduction to Artificial Immune Systems. In *Handbook of Natural Computing* (pp. 1575-1597). Springer Berlin Heidelberg.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem, Library Translation no. 1122. Ministry of Aviation, Royal Aircraft Establishment, Farnborough, Hants UK.

Resnick, M., (1994). Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds. MIT Press.

Sadik, S., Ali, A., Ahmad, F. and Suguri, H. (2006). Using Honey Bee Teamwork Strategy in Software Agents, Computer Supported Cooperative Work in Design. CSCWD. 10th International Conference on, pp.1-6.

Salem Z. Ades N and Hilali E., (2009). Algorithm Enhancement by Using Bees Algorithm, Res. J. of Aleppo Univ. Engineering Science Series No.66.

Sankoff D. and Blanchette. M., (1997). The median problem for breakpoints in comparative genomics." Computing and Combinatorics, Proceedings of COCOON `97. Lecture Notes in Computer Science 1276, Springer Verlag, New York, pp. 251-263.

Seeley, T.D. (1996). The wisdom of the hive: The social physiology of honey bee colonies, Cambridge, Massachusetts: Harvard University Press.

Seeley, T. D. and Buhrman, S. C. (2001). Nest-site selection in honey bees: How well do swarms implement the "best-of-N" decision rule? Behavioral Ecology and Sociobiology 49(5), pp. 416-427.

Seeley, T. D. (2002). When is self-organization used in biological systems? Biological Bulletin 202(3), pp. 314-318.

Seeley, T. D. and Visscher, Passino. K., (2003). Choosing a home: how the scouts in a honey bee swarm perceive the completion of their group decision making. Behav Ecol Sociobiol, (54) pp. 511–520.

Seeley, T. D., Visscher. and Passino K. M. (2006). Group decision making in honey bee swarms. American Scientist 94(3), pp. 220-229.

Socha, K., (2007). Ant Colony Optimization for Continuous and Mixed-Variable Domains. PhD**.**

Socha, K. and Dorigo, M., (2008). Ant colony optimization for continuous domains. European Journal of Operational Research 185(3), pp. 1155-1173.

Sung, H. J., (2003). Queen-Bee Evolution for Genetic Algorithms. Electronic Letters, 39(6), 575- 576.

Sholedolu, M. O., (2009). Nature Inspired Optimisation: Improvements to the Particle Swarm Optimisation Algorithm and the Bees Algorithm. Cardiff University.

Tao, G., and Michalewicz, Z., (1998). Inver-over Operator for the TSP. Proceedings of the 5th Parallel Problem Solving from Nature, September 27-30, 1998, Lecture Notes in Computer Science, pp. 803-812.]

Teodorović, D., (2008). Swarm intelligence systems for transportation engineering: Principles and applications. Transportation Research Part C: Emerging Technologies 16(6), pp. 651-667.

Tsai, H., Yang J., and Kao, C., (2002) Solving Traveling Salesman Problems by Combining Global and Local Search Mechanisms. Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 .

Uhlmann, J, (1991). Satisfying General Proximity/Similarity Queries with Metric Trees. Information Processing Letters 40, pp. 175–179

Yang X. (2005). Engineering optimization via Nature-Inspired Virtual Bee Algorithms. IWINAC 2005. LNCS 3562, pp. 317-323.

Yang, X., S., (2010) Engineering Optimization: An Introduction with Metaheuristic Applications. Wiley, Chichester .

Yianilos. P. N. (1993) Data structures and algorithms for nearest neighbor search in general metric spaces. In Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, pages 311–321, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics.

Liu Y., and Passino, K., M., (2000). Swarm Intelligence: Literature Overview, Dept. of Electrical Engineering, The Ohio State University.

Zelinka, R., Singh, S. P. and Mittal, M. L. (2010). Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches, readings on the Traveling Salesman Problem, Theory and Applications, Edited by Donald Davendra.

Zhenya, H., Chengjian, W., Bingyao, J., Wenjiang, P., and Luxi. Y., (1999). A new population-based incremental learning method for the traveling salesman problem. Proceedings of the 1999 Congress on Evolutionary Computation-CEC 99, pp. 1150-1156.

Winston, M., L. (1987). The Biology of the Honey Bee; Harvard University Press: Cambridge, MA, USA.

Wong, L., P., Low M. L., Chong, C., S., Bee Colony Optimization with Local Search for Traveling Salesman Problem.

Wolpert D.H. and Macready, W.G. (1997). No Free Lunch Theorems for Optimization, *IEEE Transactions on Evolutionary Computation*, 1 (1997), pp. 67–82.

Url- http://en.wikipedia.org/wiki/Travelling_salesman_problem.

CURRICULUM VITAE

Name Surname: Sultan ZEYBEK

Place and Date of Birth: Konya / 04.10.1989

Address: FSMVU Halic Campus Beyoğlu / İSTANBUL

E-Mail: szeybek@fsm.edu.tr

B.Sc.: YTU Mathematics Depertmant