Concentric Tabu Search Algorithm for Solving Traveling Salesman Problem (TSP)

Zeravan Arif Ali

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
January 2016
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Cem Tanova
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Ahmet Ünveren
Supervisor

Examining Committee

1. Asst. Prof. Dr. Adnan Acan
2. Asst. Prof. Dr. Mehmet Bodur
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

In this research one of the local search algorithms called the Concentric tabu search (CTS) is used to solve the traveling salesman problem (TSP). One of the well known NP-hard problems in combinatorial optimization is the TSP Problem and it is one of the most competently studied problems in the area of combinatorial optimization. Two different implementations of the Concentric tabu search (CTS): ring moves (RM) and all moves (AM) are used and compared with the traditional tabu search. For searching global optimal solutions for given TSP problems, Concentric tabu search was hybridized with Genetic Algorithm.

Computational experiments showed that Concentric tabu search gives better performance than the traditional tabu search and also improves the execution of the Genetic Algorithm (GA) for the solutions of TSP problems.

Keywords: Concentric Tabu Search, Tabu Search, Genetic Algorithm, Traveling Salesman Problem.

ÖZ

Bu araştırmada yerel arama algoritmalarından biri olan Ortak Merkezli tabu arama (OMTA) yöntemi seyyar satıcı problemini (TSP) çözmek için kullanılmıştır. NP-Zor problemlerinden biri olan TSP için en iyi çözümlerin bulunması ile ilgili literatürde pekçok çalışmalar bulunmaktadır. Ortak Merkezli tabu algoritması için iki farklı yöntem bulunmaktadır : Halka Hamle (HH) ve Tüm Hamle (TH), bu çalışmada bu iki yöntem geleneksel tabu arama ile TSP çözümleri üzerinden karşılaştırılmıştır. Ayrıca OMTA Genetik Algoritma ile birleştirilerek TSP problemlerine en iyi çözümler bulunmaya çalışılmıştır.

Yapılan deneyler ile OMTA geleneksel TS yönteminden daha iyi sonuçlar verdiği gösterilmiştir. Ayrıca Genetik Algoritma ile kullanıldığında TSP problemlerinin çözümünde Genetik Algoritma sonuçlarını iyleştirdiği gözlemlenmiştir.

Anahtar Kelimeler: Ortak Merkezli Tabu Arama, Tabu Arama, Genetik Algoritma, Seyyar Satıcı Problemini.

To Roj

ACKWNOLEDGMENT

My heartily thanks goes to my supervisor Asst. Prof. Dr. Ahmet Ünveren for his guidance, support, and patience trough out the duration of this research and for leading me to success. Without his efforts and time, this thesis would have been outright failure. I greatly appreciate it.

In the same vein, I would like to express my appreciations and gratitude to my lovely friends, Sharam, Muhammed, Qutaibah, Yazan, Sajad, and Kamal for their help. In addition, I really thank all those who have supported me to achieve my goals.

Finally, my profound gratitude to my dear family for the care, faith, continuous helps, and encourage given to me throughout my study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

The Traveling Salesman Problem (TSP) is one of the most well-known and significant combinatorial optimization problems (COP). The study of TSP is to search the minimal path to visit all cities in a given list only once a time and return to the first city. On other hand to its simple definition, solving the TSP is difficult since it is NP-complete problem and not proximal to any constant [7]. TSP can be solved with ease when there is minimum cities number, although, solving it will be very hard for big problems, required a large amount of calculating estimate time. Based on the framework of the cost matrix associated to the TSPs can be categorized into asymmetric and symmetric [4].

Tabu search (TS) algorithm explores the search space by manipulating moves to a solution in a way that fashion new solutions while trying to avoid reversing these moves for a confirmed number of iterations [3]. TS is a meta-heuristic algorithm, which can be applied for works out combinatorial optimization. TS has found its usefulness in a number of applications such as traveling salesman problem, scheduling, graph coloring, knapsack problems, etc. TS is now a reputable optimization technique and has gained high effectiveness in solving a broad range of optimization problems. Several works mentioned that TS provides best solutions to the given combinatorial optimization problem. Consequently, Tabu Search extremely used to find good solutions in particular, for large combinatorial problems [2].

Concentric tabu search (CTS) algorithm was proposed by Drezner [1] in 2002. The idea is to force the search to regions of the search space that are far away from the initial solution hoping to find a better local optimum [1]. As a variable neighborhood search approach, Concentric Tabu Search falls in the general framework of tabu search. Ring moves and All moves are two different types of the CTS which are suggested for solving combinatorial problems. In the former, search is undertaken in rings surrounding the center solution which is the starting solution, this continues from one ring (circle) to a larger one, and continues, until the search reaches a predefined radius [2]. In the latter, the rule of the search is allowed to proceed to a lower hamming distance solution under certain conditions.

Genetic Algorithm exists within evolutionary algorithm class based on natural evolution which is based on the percept of the survival of the fittest. Genetic algorithm is best heuristic algorithms that have been employed more frequently to deal with the travelling salesman problem (TSP) problems [5]. GA start with a population of different solutions to the problem, a fitness function is used for calculating individual fitness, and then a new generation will be created during the selection process, crossover and then mutation. Genetic algorithm and after finds a fitter solution the algorithm then terminate. While the algorithm continues with new population whenever the termination condition is not met [6].

Since the simple genetic algorithm convergence speed is relatively slow. A satisfied local search technique required to enhance the quality of genetic algorithm individuals before inserting them into the population and improve the local search ability [9].

In this research, Concentric tabu is hybridized with the genetic algorithm in order to maintain a balance between intensification and diversification during the search process [9].

The experimental evaluations are implemented using several benchmark problems available on TSPLIB [21], the library of TSP instances. They are tested on the local and global search algorithms and the results obtained are presented. It is noticed that computational results show that the Concentric tabu search gives promising results. Moreover, integration between CTS and GA is adequate and effective compared to implementation of the genetic algorithm alone.

The remaining part of this research is organized as follows: in Chapter 2, general gives a description of Tabu Search, Concentric Tabu Search, and Genetic Algorithm. Traveling Salesman Problem will be detailed in Chapter 3. Furthermore, the Concentric Tabu Search (Ring Moves), Concentric Tabu Search (All Moves), and Genetic algorithm with Concentric Tabu Search will be outlined in Chapter 4. While, obtained experimental results will be discussed in Chapter 5. Finally, the research will end with a conclusion and recommendations for the future works in Chapter 6.

Chapter 2

LOCAL SEARCH ALGORITHMS AND CONCENTRIC TABU SEARCH

Local Search algorithms are among accepted technique of approximate algorithms for solving combinatorial problems. The search proceed from a solution to its neighbors by implementing small move to the solution in order to discover better solutions in the search space. In this work the CTS is used as a local search procedure. Furthermore, a combination of local search algorithm represented by Concentric Tabu Search Algorithm and an evolutionary algorithm that is Genetic Algorithm are presented. It's noticed that without using local search algorithms in optimization problems, there is a less possible for the other algorithms to find the optimal solution.

2.1 Tabu Search

The Tabu Search (TS) was proposed by Fred Glover [2] in 1988. It was introduced as one of the most efficacious local procedures for solving combinatorial optimization in a number of areas such as traveling salesman problem (TSP). It is usually obtained by altering one solution to get the next (better solution) according to some neighborhood structure. Likewise, TS has a faster execution speed than other local search procedures because it does not revisit already seen solutions, considering them tabu. This is possible because each move is recorded to avoid revisiting already seen solutions.

Tabu search procedure begins from a starting solution, and at every step to hopefully enhance the objective criterion value such a move to a neighboring solution is chosen. This is convenient to a local improvement procedure except for the fact that a move to a solution worse than the current solution may be acquires [8]. Figure 1 shows the flowchart of the tabu search.

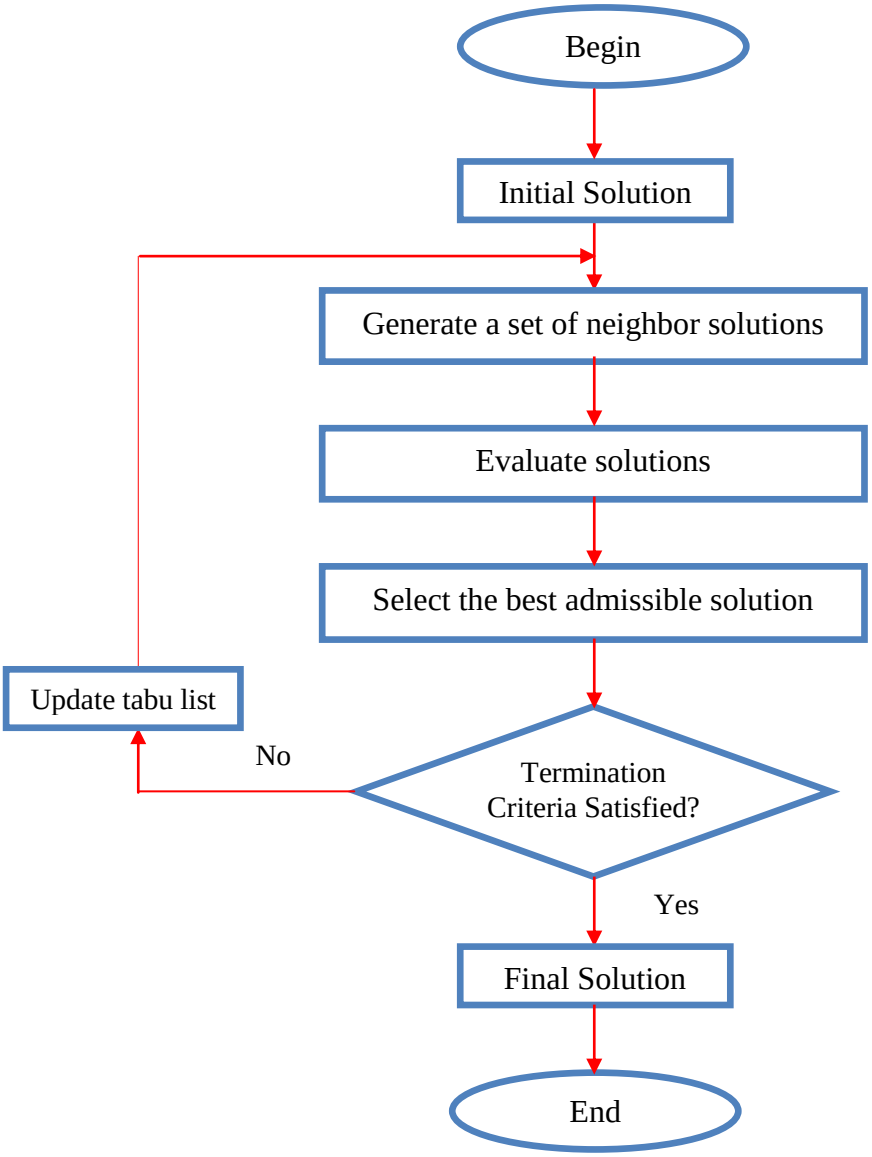

Figure 1: Tabu Search FlowChart

2.1.1 Tabu Search Parameters

In the following parts, we will describe in details the elements of the Tabu Search algorithm:

2.1.1.1 Short Term Memory

Record a set of solutions latterly investigated to be discouraged in order to prevent revisiting an already seen solution. If hidden solutions show up on the tabu list, it cannot be returned until it reaches a termination point. It is used to prevent a search from becoming trapped in local minima.

2.1.1.2 Long Term Memory

This memory keeps characteristics of better solutions which will be employed in:

- Intensification: attaching preference to certain characteristics of a group of more promising solutions.
- Diversification: discouraging the characteristics of choices solutions so as to diversify the search to other regions of search space.

2.1.1.3 Move

Tabu search moves from one solution to another, been an improvement heuristic in search of a more promising solution. The technique of transitioning from one solution to another is predefined by a set of rules which is known as a move. The neighborhood of the current solution is the series of whole solutions that can be achieved from the solution using a pre-specified move.

2.1.1.4 Tabu List (TL)

To prevent revisiting already seen solutions, TS employs a tabu list in which tabu moves or characteristics are listed. Moreover, the word tabu is coined from this list of prohibited moves. Tabu lists with short length may not stop cycling outcomes in

information loss while on other hand, tabu lists with long length may overmuch expand neighborhood so that moves are fixed to some reach. In essence, if the tabu list is too short it leads to deteriorating the search results. In contrast, if the size of tabu list is too long this means it cannot effectively prevent cycling.

Intensification of the search used to decrease the tabu list size whereas diversification tries to increase the size of the tabu list and penalize the frequent move.

2.1.1.5 Aspiration Criteria

Tabu constraints are subject to an important omission. In a situation where a tabu move has a sufficiently better assessment where it can be evaluated to a solution better than any seen yet, then its tabu categorization may be overruled. Aspiration criterion is the rule that allows such an exception to exist [2].

Aspiration criterion which is frequently used is reverting to a solution better than the last found solution so far.

2.1.1.6 Stopping Criterion

Some prompt stopping conditions are:

- A given number of fixed iterations.
- A given amount of Processor time.
- No feasible moves into the locality of the current found solution.
- Evidence can be given than an objective functions output is feasible.

2.2 Concentric Tabu Search (CTS)

Concentric tabu search (CTS) was proposed by Drezner [1] in 2002. The underlining idea is to push the search to parts of the solution space which are far away from the center solution with the hope of finding a better local optimum. For example, the distance (Hamming) between two solutions is the number of different variables with

different values [2]. The TSP seeks the best permutation of assigning between cities. For the TSP, the hamming distance use to find the number of cities that are assigned to different sites between best found solution and its neighbor. Furthermore, fitness value of tour that is assigned to cities (or the total length of the tour).

The fundamental principle of the Concentric tabu search includes: A starting solution called the center solution of the search space. The neighborhood of every individual solution is defined in similar way as in standard tabu search [2]. In the specific case of the TSP, the neighborhood consists of all possible exchanges among cities. The search is performed in rings (circles) around the starting solution. If a solution which is better than the best found solution is detected, it replaces the starting solution and the search continues with that solution. Otherwise, the search makes a start by calculating solutions which are farther away from the starting solution. For any problem there is a fixed number of iteration, and once it is reached, the algorithm terminates [1].

2.2.1 Moves in Concentric Tabu Search (CTS)

Two different types of CTS are proposed, these are: ring moves and all moves.

2.2.1.1 Ring Moves (RM)

The search is executed in rings (circles) around the starting solution, proceeding from smaller ring (circle) to a larger one, and this continues, until a pre-specified radius is reached. Randomly solution is selected which is a starting (center) solution. We keep three solutions, the first one contains the center solution and the other will fill under the specific condition, their aim is to forcing the search away from the center solution. Details are discussed in chapter 4.

2.2.1.2 All Moves (AM)

In this approach, the concept of RM is replaced by another technique. A list of the best seen solutions is preserved (a list contains only the starting solution at the beginning). A set of members, whose surrounding solutions were not tested, is flagged. The iteration stops whenever there is no member in the list is flagged. See more details in chapter 4.

2.3 Genetic Algorithm (GA)

Genetic Algorithm (GA) was first proposed by the American Scientist John Holland in 1975 [5]. In artificial intelligence, GA is a search heuristic that imitates the survival of the fittest. It is one of the evolutionary search approaches which can give optimal or near to optimal solutions to combinatorial optimization.

The underlining principle of GA is to produce a random initial population of solutions to the problem, called Chromosomes, and then enhances this population of solutions after some iterations known as Generations. Throughout every generation, the quality of each chromosome is calculated, by applying some measurements of quality (fitness). To produce the next generation of chromosomes, offspring are inspired by either modifying a chromosome using a mutation operation or mixing of two chromosomes from present generation that is using a cross over operation or doing both operations. A new set of solutions (generation) is created by selection operation, based on the quality values of some of the offspring and parents, and refusing other individuals so as to maintain a fixed size of the population. Fitter individuals have better chances of been selected for reproduction. After so many iterations, the GA converges to the fitter chromosome, that hopefully contains the

optimum or near to optimum solution [6]. Figure 2 illustrates the flowchart of the Genetic Algorithm.

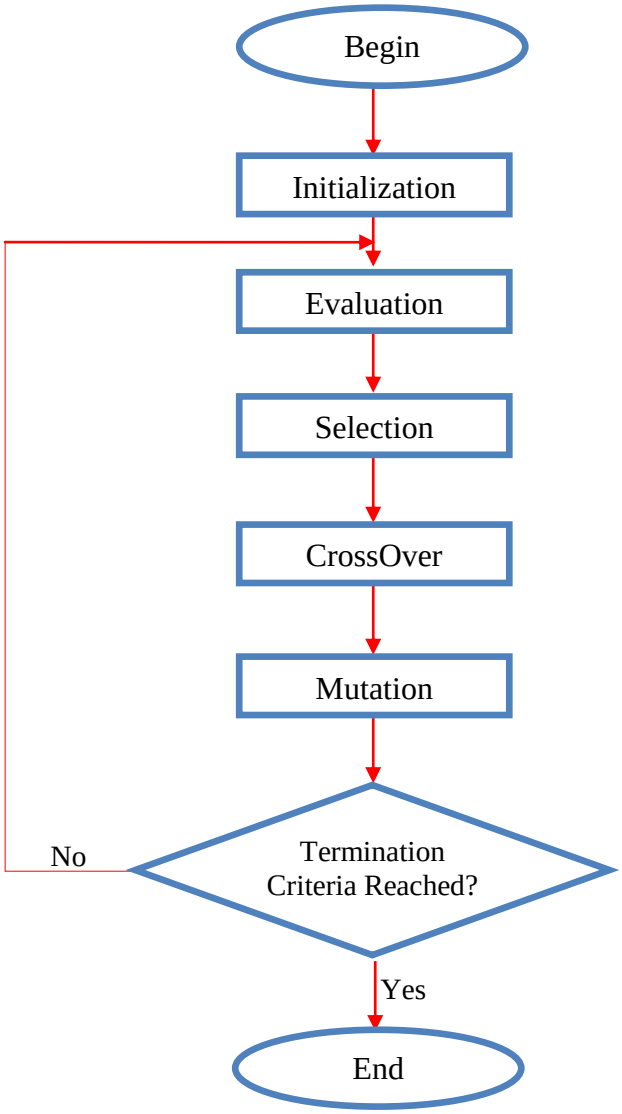

Figure 2: Genetic Algorithm Flowchart

2.3.1 Genetic Algorithm Parameters

In the following parts, we will describe in details the components of the Genetic algorithm:

2.3.1.1 Representation

Chromosome representation is the simile between the real world and the evolutionary algorithms nature. This is shown in the connection between genotype and phenotype, where the encoding of the individuals within the evolutionary algorithms is called genotype while the phenotype is the feature of an individual resulting from its interactions with the environment. In order to obtain the minimal tour for a predefined list of m cities using GAs, the path representation is more natural for TSP [9]. For instance, suppose {1,2,3,4,5} are labels of nodes in a five nodes instance, then a tour {4-3-2-1-5-4} may be represented as (4,3,2,1,5).

2.3.1.2 Evaluation/Fitness Function

Also known as the fitness function is a measure of a solution's quality. The use of evaluation function is to evaluate if an individual solution is good, then what is the level of goodness? The length of tour is the quality of an individual solution in the TSP. The fitness quality is calculated during the creation of an individual as shown in equation 2.1. After each individual is created its quality is evaluated [10].

$$Fitness_{\ chromosome} = \frac{1}{\sum_{i=1}^{n} ti} \qquad \text{Eq. 2.1}$$

Where,

n = total numbers of cities.

t = distance between two cities.

2.3.1.3 Population

The generated set of individuals in a given generation of an optimization problem which contains a fixed number solution is known as the population. The initial population for TSP is formed by random permutation of the cities.

2.3.1.4 Selection Operation

This is the technique used to choose the chromosome whose fitness value is good or better in order for the fitter individuals be parents of the next generation [10]. Consequently, the high quality individuals get a better chance to be parents as compared to individuals with low quality.

2.3.1.4.1 Tournament Selection Mechanism

In the Tournament selection technique a set of chromosomes (k) are selected for which more qualitative individuals are selected as parent. The number of chromosomes in the tournament is equal or less than the size of the population [11]. Figure 3 shows the steps of the tournament selection.

Step **1**: Randomly picking a point within the population,
Step **2**: Choose as many individuals as outlined by the size of the tour =*k*,
Step **3**: Arrange the individuals according to fitness,
Step **4**: Select the fittest two to be parents.

Figure 3: Steps of the Tournament Selection Mechanism

Individuals having the best fitness functions out of the k tournament contestants have better chances of being selected. This type of selection gives definite chances for not already chosen individuals to be chosen.

2.3.1.5 Crossover

New chromosomes are recombined so as to determine the parents for the next generation after the completion the evaluation process. The most effective step for this procedure is called crossover [11]. If there is no recombination, parents and offspring are same. If there is recombination, offspring bringing about better parts than that of parent chromosome. If the rate of crossover is exact, then offspring are all brought about by using crossover operation. On the other hand, where there is none, all new individuals are created from exact copies of individuals from the old generation [10]. Crossover operation, tries to generate new chromosomes in order to obtain better parts from old individuals and perhaps the new individuals will have better quality. Ordered crossover operation is applied in this work to a pair of chromosomes.

2.3.1.5.1 Ordered Crossover

Two crossover points are randomly selected in this operation on two randomly selected parents. The genes of the selected parents between the cut points are transferred to the offspring. The genes which are not filled in an offspring are obtained from the other parent starting from the second crossover point and transferred to the offspring in the order they emerge.

As shown in Figure 4 below, in offspring C1, since genes C, D, and E are obtained from P1, we obtain genes B, G, F, and A from P2. Starting from the second crossover point, which are to say that for the sixth gene, we copy genes B and G as the sixth and seventh genes respectively. We then turn around and copy genes F and A as the first and second genes respectively.

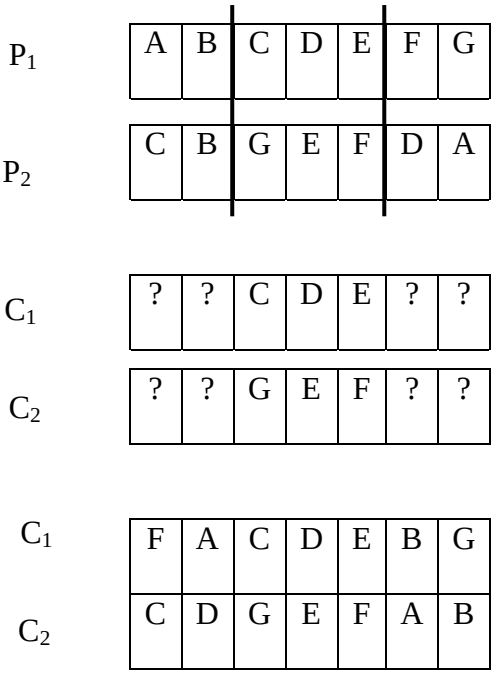

Figure 4: Ordered Crossover

2.3.1.6 Mutation

This operation allows new individuals to be generated by choosing individuals with better fitness value from the population. After crossover is performed, mutation operation is done. During this process, an individual in the current population is randomly taken, switched and mutated [15].

If there is no mutation, it means there is no change in offspring taking place after crossover. If mutation operation is performed, a part of the individual is changed. If mutation rate is exact, whole individual is mutated; while if there is none, nothing is mutated [10]. Mutation operation is applied to avoid plunging genetic algorithm into local minima, but it should occur rarely, because then GA will in effect shift to random search. Increasing mutation rate leads to gains in average option. It also leads to considerable improvement in result with very little cost. Nevertheless, it should happen less repeatedly. In this work, reverse Order mutation is applied to the pair of chromosomes.

2.3.1.6.1 Reverse Sequence Mutation

In this approach, we select a series of *X* restricted by two randomly chosen points y and z, where y<z. Like crossover operation the gene order in the sequence will be reversed. Figure 5 below shows an example implementation of this operation [15].

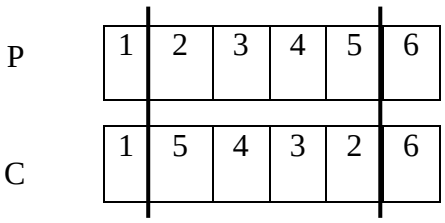

Figure 5: Reverse Sequence Mutation

2.3.1.7 Termination Condition for Genetic Algorithm

Due to the fact that GA is a non-deterministic search approach, this makes it hard to properly determine the convergence of the algorithm. Moreover, the fitness of a population may stay fixed for a number of iterations since a better individual is not created. This implies the application of typical termination criteria becomes questionable. A prevalent used technique is to end the GA after a fixed number of iterations and then compare the fitness of the fittest individuals of the population with the problem determination. If no adequate individuals are found, the Algorithm might perhaps be restarted or a different search technique is used.

Chapter 3

TRAVELING SALESMAN PROBLEM

3.1 Travelling Salesman Problem (TSP)

The TSP is the most famous combinatorial optimization. It is about finding the path of a salesperson who gets started from a location (initial location), visits a specific number of other locations and reverts back to the same starting point in such a fashion that the overall distance covered is minimum and every location is visited precisely once [13].

The idea of TSP originated from the Swiss mathematician Euler. TSP originated from his work 'Studied the Knight's tour Problem' which he conducted in 1766. A significant amount of work was conducted in the 18^{th} century after Euler, by Hamilton W. from Ireland and Penyngton T. from Britain respectively. Their study was about finding paths and circuits on the graph of the dodecahedral, favoring many conditions. A vast majority of the works on TSP were done from 1800 to 1900. Consequently, Lawler, Shmoys, Lenstr, Kan and Rinnoy didn`t say anything regarding TSP. Then, M. Flood presented results relevant with TSP in the year 1940. After that Fulkerson, Dantzig, and Johnson found an approach for working out TSP in the year 1950. They showed the performance of the technique by working out a forty-nine city problem. Nonetheless, it seems noticeable, in the middle of 1960's that the TSP was not able to be worked out in polynomial time by using LP. Consequentially, this category of problems became common as NP-hard (Non-

Polynomial Hard) problems. Considerable advancement took place in the late 1970s and early 1980s, when Padberg, Grötschen, Rinaldi and others achieved to work out a TSP example with up to two thousand three hundred and ninety two (2392) cities, applying branch-and-bound cutting planes techniques.

Later, TSP turns into an attractive combinatorial optimization problem and a lot of findings applied it as standard yardstick problem for heuristics. This problem is highly important for many experimental areas like operational research (OR) and computer science (CS). TSPLIB is the library of benchmark model examples for the traveling salesman problem which was published in 1991. After that, the library is used to compare available results on it with the outcomes of algorithms done by researchers. Accordingly, benchmark problems available in TSPLIB are used in this thesis. Results are compared as well.

3.1.1 Definition

The traveling salesman problem is defined as:

TSP = (G, f, t): G = (V, E) an entire graph,

Where, f is a function (V x V) → Z, $t \in Z$,

The graph G is outlines a traveling salesman's travel cost that doesn't override *t*. Figure 6 shows a TSP instance for initial cities.

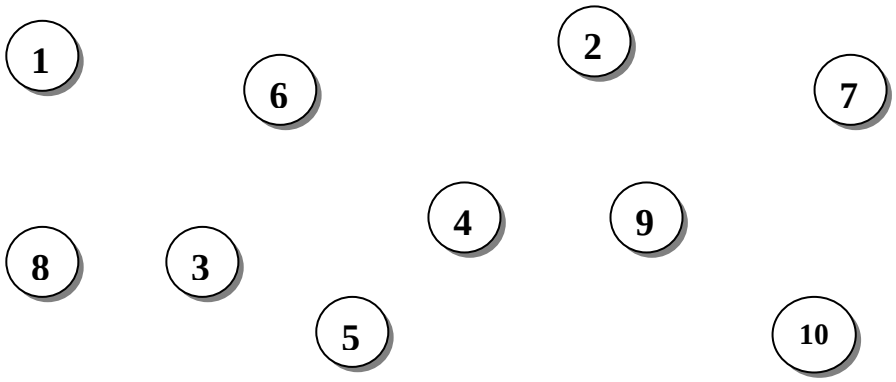

Figure 6: Sample for locations of the given placement of the cities in the map

The distance between cities is calculated by using the Euclidean Distance as follows:

$x = (x_1, x_2, x_3 \ldots x_n)$,

$y = (y_1, y_2, y_3 \ldots y_n)$.

$$D_{xy} = \sqrt{(x-y)^2} \qquad \text{(Eq. 3.1)}$$

{ D_{xy} is the distance between coordinate *x* and coordinate *y*,

x are locations of the city on coordinate *x*,

y are location of the city on coordinate *y*,

Table 1: Distance Matrix for ten cities

0	107	241	190	124	80	316	76	152	157
107	**0**	148	137	88	127	336	183	134	95
241	148	**0**	374	171	259	509	317	217	232
190	137	374	**0**	202	234	222	192	248	42
124	88	171	202	**0**	61	392	202	46	160
80	127	259	234	61	**0**	386	141	72	167
316	336	509	222	392	386	**0**	233	438	254
76	183	317	192	202	141	233	**0**	213	188
152	134	217	248	46	72	438	213	**0**	206
157	95	232	42	160	167	254	188	206	**0**

The above distance matrix is used to calculate the length of a tour. The mentioned matrix sample is for TSP instance with 10 cities. Figure 7 shows a possible tour in a TSP with ten cities.

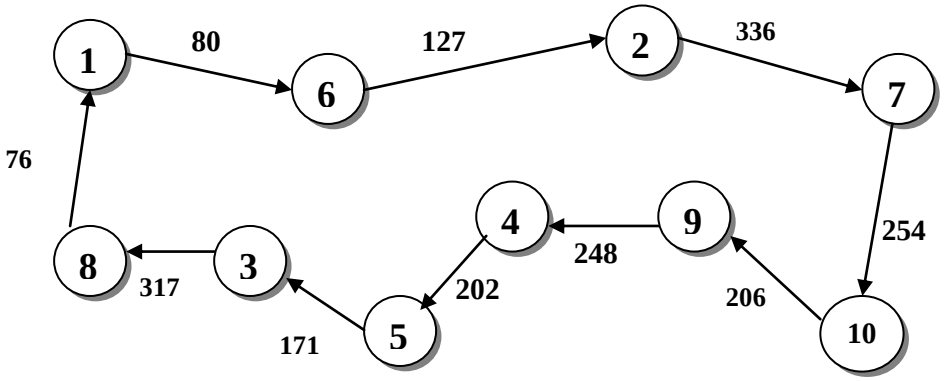

Figure 7: Sample tour with edge distances for Ten Cities

To calculate the tour length, assume P = {p_1, p_2, p_3… p_n} as a feasible tour for n cities:

$$L = (\sum_{i=1}^{n-1} D_{ci\ ci+1}) + D_{ci+1\ c1} \qquad \text{(Eq. 3.2)}$$

The problem mystery in finding a lower or minimal path transcend through all points once. E.g. the Path1 {1, 6, 2, 7, 10, 9, 4, 5, 3, 8, 1} transcending all the points by calculating the distance between cities and sum of those based on the distance matrix. Path1 cost is 2017.

TSP is categorized into two classes Symmetric TSP and Asymmetric TSP according to the types of graph and arrangement of distances.

- **Symmetric Traveling Salesman Problem:** in STSP, the distance between two points is always the same in both directions and the cost of moving from point *a* to point *b* is equal to the cost of moving from point *b* to point *a.* Figure 8 and figure 9 below show an instance and the distance matrix regarding symmetric.

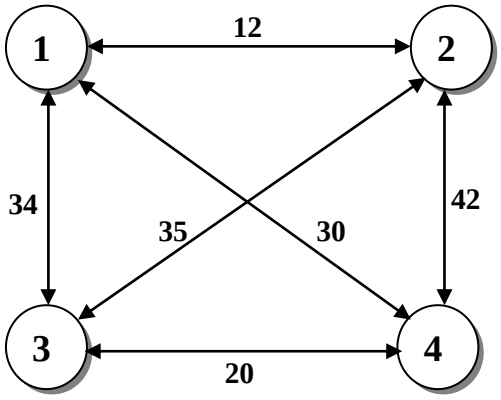

Figure 8: Example of symmetric TSP

	1	2	3	4
1	**0**	12	34	30
2	12	**0**	35	42
3	34	35	**0**	20
4	30	42	20	**0**

Figure 9: Distance matrix for figure 8

- **Asymmetric Traveling Salesman Problem:** in ASTSP, the distance between two points is not the same or paths may not found in both directions. That is the cost of moving from point *a* to point *b* is not equal to the cost of moving from point *b* to point *a*. Figure 10 and figure 11 below show an instance and the distance matrix regarding asymmetric.

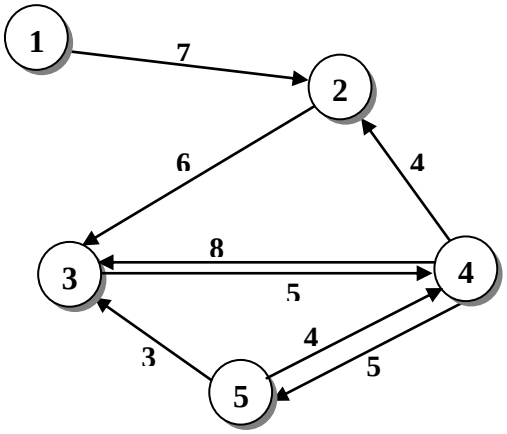

Figure 10: Example of Asymmetric TSP

	1	2	3	4	5
1	**0**	7	0	0	0
2	0	**0**	6	0	0
3	0	0	**0**	5	0
4	0	4	8	**0**	5
5	0	0	3	4	**0**

Figure 11: Distance matrix for figure 10

The Symmetric traveling salesman problem samples from TSPLIB are taken and applied in this work. There are different techniques used for solving the traveling salesman problem. Practically, these techniques are divided into two groups: exact and approximation algorithms.

- **Exact algorithms**: are approaches which use mathematical techniques such as Lagrangian Relaxation, Branch and Bound, Integer Linear Programming etc.
- **Approximation algorithms:** are approaches which use heuristics and insistent progresses in the problem solving process. These are classified into two Constructive heuristics and Improvement heuristics such as Greedy, Heuristics, Nearest Neighborhood, Tabu Search, Evolutionary Algorithms, Ant Colony optimization, etc.

3.2 Related Works for TSP

Since TSP is a field of combinatorial optimization and an NP-complete problem, there is no precise approach to solve this problem and obtain good results. A lot of algorithms are used to solve travelling salesman problem. Some of them have optimal solutions, while another's have the near to optimal solutions. There are different heuristics methods which are used to explore the solution space for TSP.

A. Arananayakgi [14] the work proposed a solution to the travelling problem using genetic algorithm GA operators to minimize the overall distance and time. It is done by generating the fittest criteria using selection operation, crossover operation and mutation operation. The purpose of the proposed approach is to create fitter solutions in acceptable time. Consequently a new crossover approach, the Sequential Constructive Crossover method is used.

Krishna, Ravindra , Gajendra [17] defined a rewarding method for working out the traveling problem. They used an enhanced heuristic algorithm Ant Colony Optimization. This work studies the precocious convergence and stagnation prevention by using initial ants' distribution strategy and effective heuristic parameter updating according toentropy.

Thamilselvan, Balasubramanie [18] deals with TSP. both Genetic Algorithms and Tabu Search were tested separately and results were compared. After that, the two algorithms were combined together and the resulting performance was compared.

Fiechter [19] for solving the TSP a parallel tabu search was described. The memory concept was used as a work as well as a new approach of move. The outcome argues the efficiency of the algorithm in obtaining a near optimal solution to huge problems.

For working out traveling salesman problem, the researchers applied different approaches and various local search algorithms. Furthermore, local search techniques usually adapted to enhance the outcomes in most works. This has been done by reason of the response of local search algorithm in combinatorial optimization problems.

Chapter 4

CONCENTRIC TABU SEARCH FOR SOLVING TRAVELING SALESMAN PROBLEM

4.2 Concentric Tabu Search Algorithms for TSP

This chapter gives an insight to the proposed work for solving the travelling salesman problem by Concentric Tabu Search algorithm. Furthermore, a combination of local search algorithm represented by Concentric Tabu Search Algorithm and an evolutionary algorithm that is Genetic Algorithm are presented.

4.1.1 Finding initial/center solution

To implement a Meta heuristic Concentric Tabu Search algorithm basically we need to have an initial or center solution and it can be generated randomly. Throughout using Concentric Tabu Search algorithm for traveling salesman problem, selecting the initial feasible solution is one of the significant steps for obtaining an acceptable solution. CTS algorithm depends on the selecting of the initial solution. The Concentric Tabu Search moves in the direction of the selected original solution considering the neighborhood of the center solution to improve the search.

4.1.2 Calculation of the fitness value

Euclidean Distance formulation was described in Eq. 3.1 and Eq. 3.2. The distances between cities are calculated to obtain the tour length that salesman travelled. Distance matrix created by providing node coordination and by applying Euclidean formulation and the calculation of the total distance is extremely easy.

4.1 Algorithms Description

This section illustrates the two various types of CTS that explained in the previous sections. Details below discuss how the algorithms are implemented to the travelling salesman problem. The figure 12 illustrates the principle of Concentric tabu search while the figure 13 show the flowchart of the Ring Move algorithm for TSP.

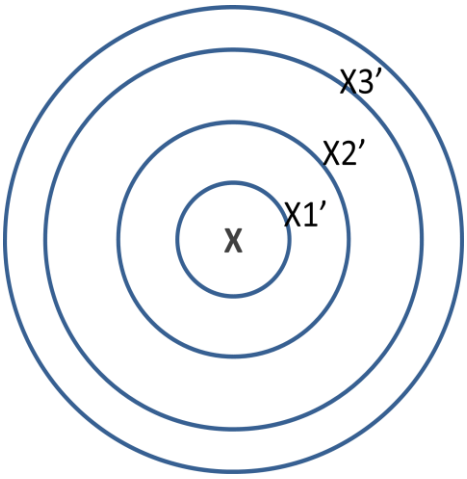

Figure 12: Concentric Tabu Search Principle

The figure above shows that the search is performed in rings around the starting solution. If a solution which is better than the best found solution is detected, it replaces the starting solution and the search continues with that solution. Otherwise, the search makes a start by calculating solutions which are farther away from the starting solution. For any problem there is a fixed number of iteration, and it is reached, the algorithm terminates.

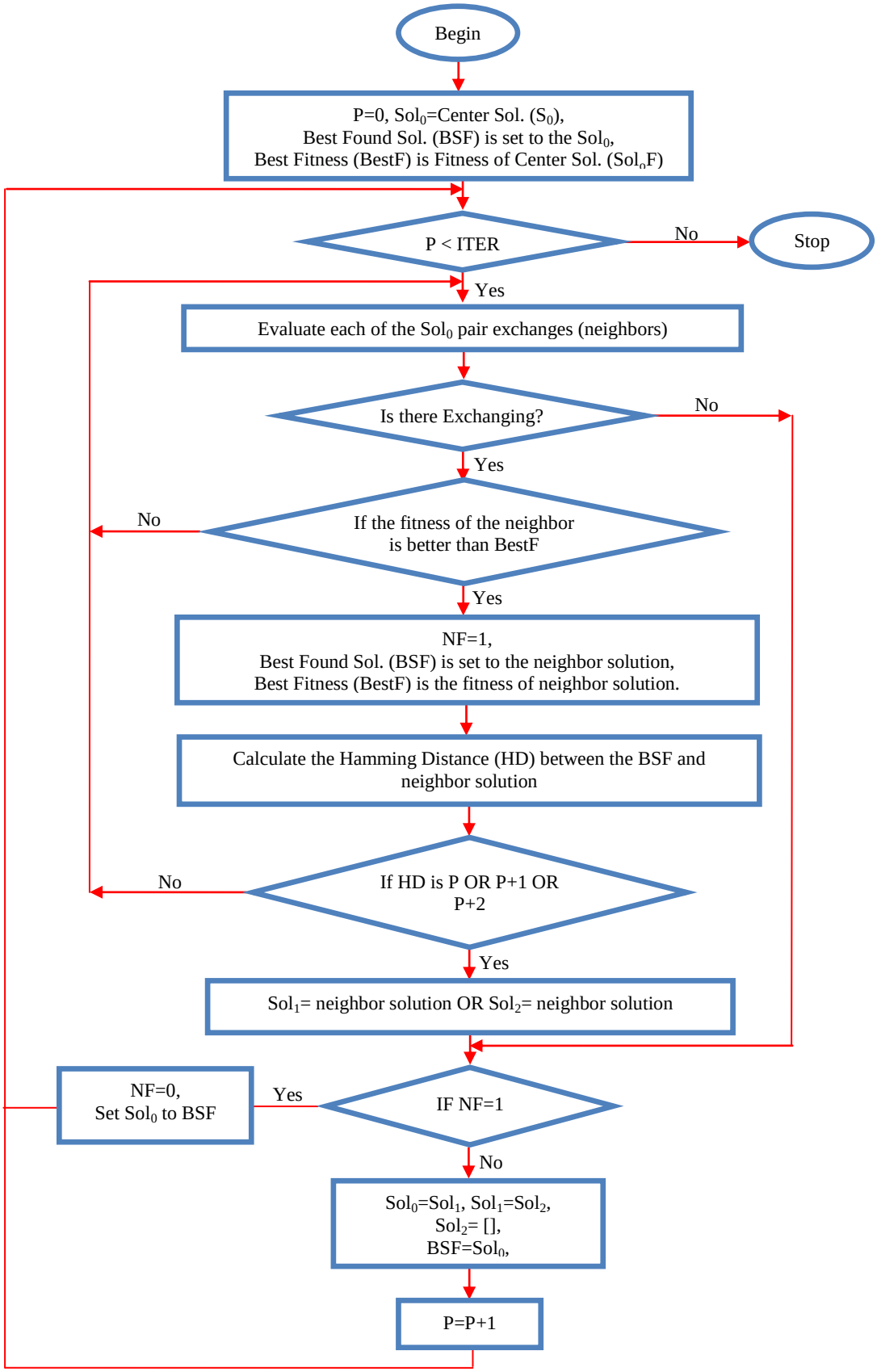

Figure 13: Ring Move Algorithm Flowchart

As it can be seen in Figure 13 there is a center solution randomly selected as a starting solution and it is the best found solution following that we have while loop which iterates through the Concentric tabu search, furthermore, there are three solutions which are Sol_0, Sol_1, and Sol_2. Sot_0 set to the center solution and the rest are empty.

The pair exchanges (neighborhood) are evaluated by swapping cities. For example TSP instances for 5 cities (1, 2, 3, 4, 5). S_0 (center solution) may be {2, 4, 5, 3, 1} and has a fitness value (total sum of the tour). Figure 14 shows the exchanges pairs of the center solution.

Figure 14: Exchange Pairs Example between Two Solutions

If the fitness of an exchanged solution (neighborhood) is less than the fitness of the center (S_0), the exchanged solution will be the best solution and the remaining neighbors are comparing to the best solution. If the hamming distance of an exchanged solution is *P*, it is ignored and the rest of exchanges are evaluated. On the other hand, if the hamming distance is *P+1* or *P+2*, Sol_1 or Sol_2 are updated when needed. It is noteworthy that the original Sol_0 is still used for the rest of the pair exchanges. Whenever the new best found solution found by examining all the neighbors of Sol_0, the center solution is set to the best found solution and the loop will continue. Once all solutions in Sol_0 are exhausted, move Sol_1 to Sol_0, Sol_2 to Sol_1, and clear Sol_2. Contrarily increase the counter, whenever stopping condition not met, return to the loop. The figure 15 illustrates the flowchart of the All Move algorithm for TSP.

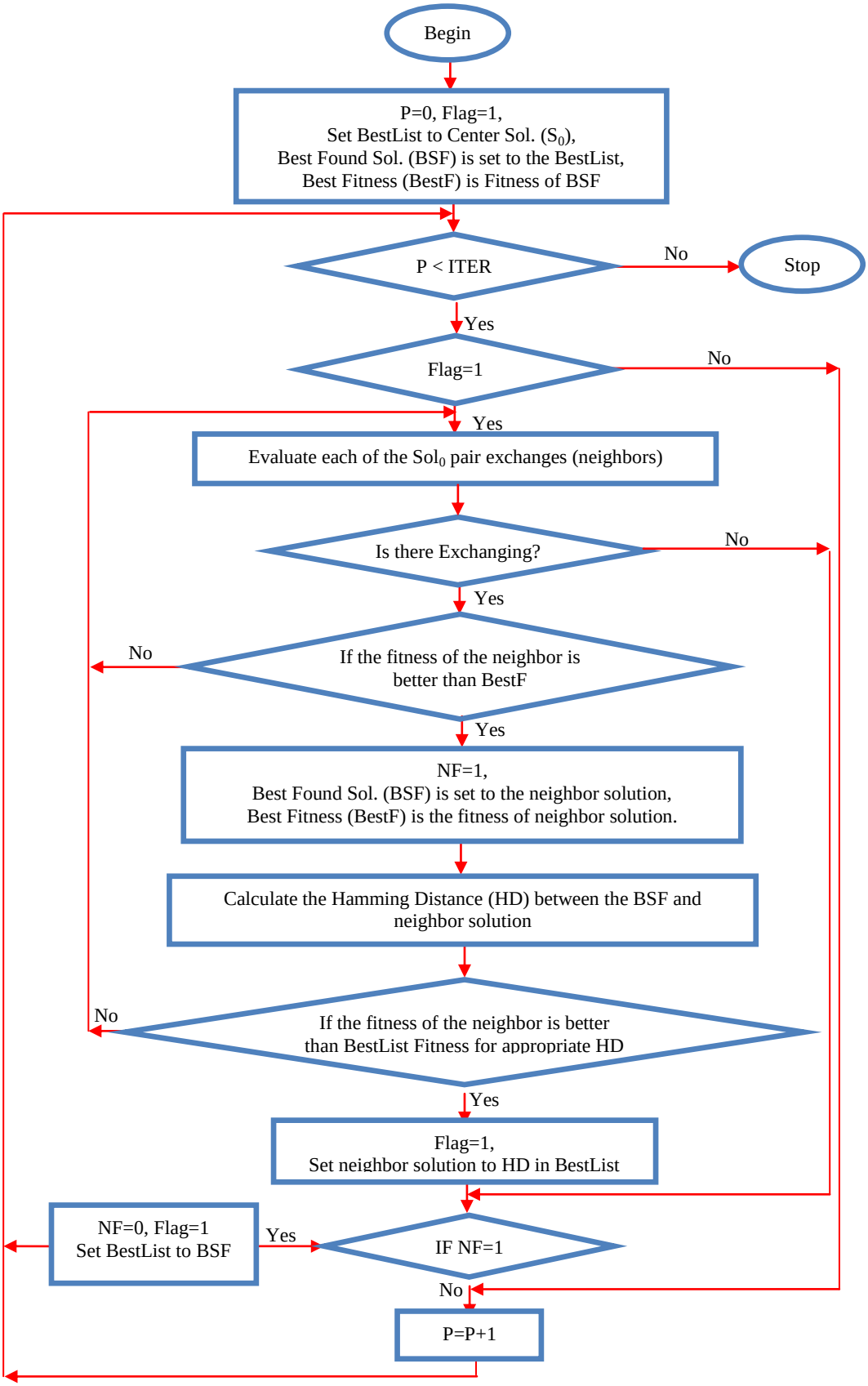

Figure 15: All Move Algorithm Flowchart

Figure 13 describes the steps of the All Move algorithm. As we can see there are no lists like in the Ring Move algorithm. A different technique is applied. Here, the list of best encountered solutions is only containing the center solution at the beginning and it is flagged. In the loop a flagged solution is checked, if it does not exist, then the algorithm will terminate with the current solution. Otherwise the flagged solution will be taken for exchanging pair wise and its flag is changed to zero. Like the Ring Move, a comparison between the fitness of the center solution and fitness of exchanged pairs is done, whenever the fitness of the neighborhood is better, the best one is updated and the remaining are evaluated in the same manner. Assuming that the exchanged one is fitter than the best encountered solution for the appropriate *P*, it replaces it and flagged. Where a new best found solution is found by examining all the neighbors of the selected solution, the center solution is set to the new best found solution and the loop is continued until stopping condition met. AM algorithm allows moves to a ring with smaller *P*; if they improve the best encountered value of the objective function for that *P*. Figure 16 shows flowchart of Genetic Concentric Tabu Search Algorithm.

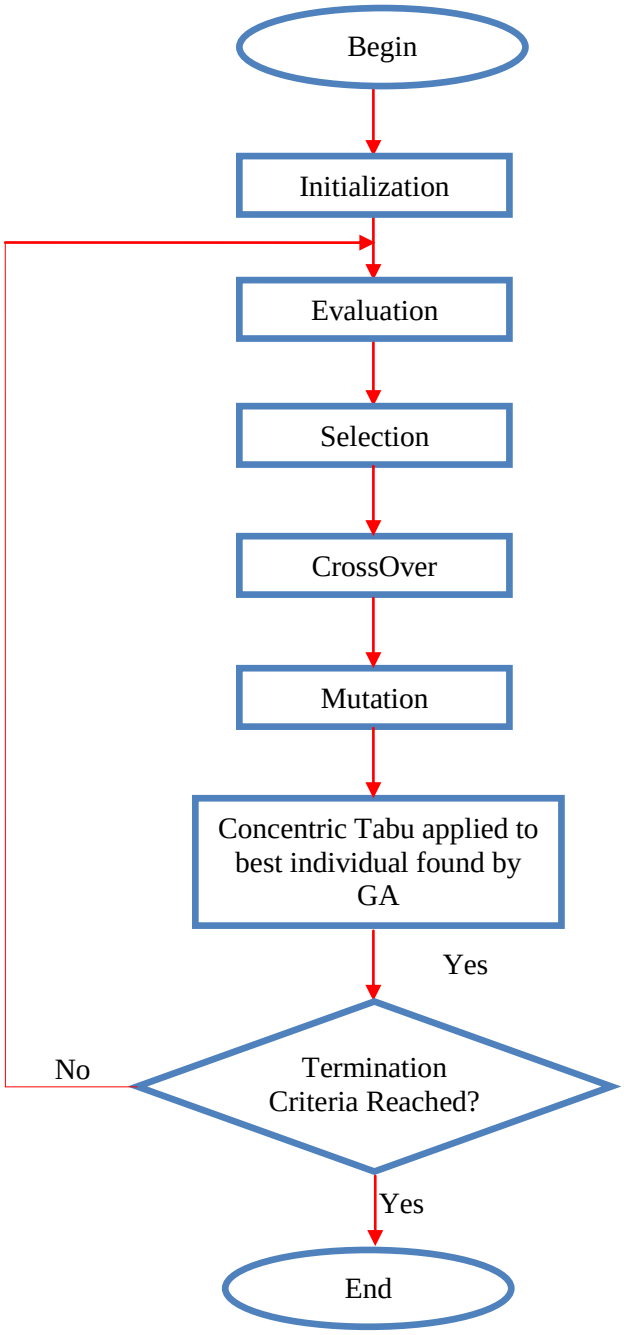

Figure 16: Genetic Concentric Tabu Search Algorithm Flowchart

In this algorithm, tournament selection is used as an operator for the selection. Tournament selection has been described in Chapter 2. Furthermore, ordered crossover has been used as discussed in section 2.4.1.5.1. Finally, reverse ordered mutation has been applied as a mutation operator and it's explained in Chapter 2. Pending the procedure, randomly generate the initial population. Consequently,

every individual in the population is evaluated. Whenever the termination rule is not met, individuals are selected for crossover and mutation operation. Fitter individual in the population is passed to local search (CTS) and new population is created.

This is the Genetic Concentric tabu search algorithm heuristic which is an incorporation of Genetic algorithm GA and Concentric Tabu Search CTS techniques which are explained in Chapter 2. Moreover, local search algorithm, CTS is applied to the best individual found by Genetic algorithm to preserve a balance between both exploitation and exploration during the search process. GA improve whole the population and the purpose of CTS is to improve on best solution. In combinatorial optimization, the requirement for local search techniques is significant in order to get better results, considering there is no guarantee for optimal solution without a local search algorithm. As a result, the local search algorithm by generating limited moves over the given solutions scheme obtain better solution than the given solution. Accordingly it's noticed that without using local search algorithms in optimization problems, it is less possible for the EA to get the optimal solution.

Chapter 5

EXPERIMENTAL RESULTS

5.1 TSP Problems

Computational results for the TSP are explained in this chapter. Experiments are obtained by using fifteen symmetric TSP benchmark problems which are as follows: kroE100, kroD100, kroC100, kroB100, kroA100, kroA150, kroA200, KroB150, kroB200, Berlin52, Bays29, Eil101, Lin105, Ch150, and Rat195. These are accessible from TSPLIB. Furthermore, each problem is tested ten times to know the capability of algorithms.

The structure and contents of the above symmetric are not different. The name of the problem is given in the first line of the problem description file, like Rat195. The type of the problem is given in the second line which is TSP. the next line give a comment of the problem followed by the information regarding the dimension line. For the problem which was mentioned above the dimension is 195. Later edge weight type is set, that is for example EUC_2D. Finally, the coordinates of cities are presented followed by EOF.

5.2 Results for TSP

Computational results are evaluated according to the components of algorithms which are discussed in the following parts.

5.2.1 Result for Tabu Search (TS)

Table 2 below describes the results obtained for 15 problems available in the TSPLIB. Each problem was solved ten times. Results for all problems have been done by local search algorithm using tabu search.

Table 2: Result of the Tabu Search Algorithm

#	Problem	Optimal	Best	Worst	STDEV	Error	Average	Time
1	KroA100	21282	29381	36567	2202	0.3805	31222	16.78
2	kroA150	26524	40887	46841	1672	0.5415	42860	20.11
3	kroA200	29368	51884	59385	2636	1.4379	55661	25.03
4	kroB100	22141	30095	33181	1687	0.3078	30445	16.20
5	kroB150	26130	39668	46965	2414.2	0.5181	43885	18.81
6	kroB200	29437	51390	57162	2132	1.4147	54448	24.72
7	kroC100	20749	29333	32511	1021	0.4137	31181	17.70
8	kroD100	21294	29054	32114	964	0.3644	30752	17.65
9	kroE100	22068	30381	34419	1266	0.3766	31935	17.88
10	Bays29	2020	2043	2088	11	0.0113	2068	9.24
11	Berlin52	7542	8175	9513	455	0.0839	8854	8.728
12	Eil101	629	764	831	19	0.2146	791	10.66
13	Lin105	14379	20302	26371	1838	0.4119	22627	10.15
14	Ch150	6528	9435	10069	182	0.4453	9924	12.36
15	Rat195	2323	3686	4257	185	0.5867	3958	14.76

Table 2 illustrates the outcomes of the tabu search technique. In the second column of the table above, name of the problem is given, followed by its optimal solution. The third column gives the best solution obtained by applying tabu search algorithm. Next column is the worst solution obtained during ten times run of the algorithm. Fifth column shows the standard deviation which is a measure of how wide values are dispersed from the average value while next column is a percentage of excess of the best solution and average solution over the optimal solution of ten runs. The column coming after that is the average solution over the best sol., while the last column is average of time of execution (in second) by the algorithm.

5.2.2 Result for Genetic Algorithm (GA)

Table 3 below describes the results obtained for 15 problems available in the TSPLIB. The experiments were executed ten times for each problem. Initial population was generating randomly. Population size is 200, crossover probability is 1.0 (i.e., 100%), mutation probability is 0.01 (i.e., 1%), and 3000 generations was set as the termination criterion.

Table 3: Result of the Genetic Algorithm

#	Problem	Optimal	Best	Worst	STDEV	Error	Average	Time
1	KroA100	21282	22562	25045	925	0.0601	23722	161.3
2	kroA150	26524	29232	31833	834	0.1020	30328	829.7
3	kroA200	29368	41052	47364	1872	0.3978	42721	2536
4	kroB100	22141	23698	25193	513	0.0703	24242	185.8
5	kroB150	26130	29770	31827	702	0.1393	30981	744.7
6	kroB200	29437	39704	45110	1644	0.3487	41543	2481
7	kroC100	20749	22706	24661	683	0.0943	23667	183.6
8	kroD100	21294	23164	24599	577	0.0772	23516	187
9	kroE100	22068	23532	26142	738	0.0663	24442	191.2
10	Bays29	2020	2020	2182	46	0	2096	2.529
11	Berlin52	7542	7544	8534	331	0.0002	8116	22.8
12	Eil101	629	685	743	18	0.0893	708	184.2
13	Lin105	14379	14632	16591	512	0.0195	15672	221.7
14	Ch150	6528	7313	7911	193	0.1251	7654	191
15	Rat195	2323	3134	3402	91	0.3418	3238	2305

Table 3 gives the results of fifteen symmetric instances available on TSPLIB of size from 29 to 200. The goodness of the solution is precise to the total of runs. Only one problem, Bays29 of size 29 could be solved exactly at least once in ten runs within reasonable time using genetic algorithm. In table 3, the column ‘Problem’ refers to the problem name in TSP library; the column “Optimal” indicates the optimal solution available in TSP library; the columns ‘Best’, ‘Worst’, ‘STDEVP’ and ‘Average’ present the best one, worst one, standard deviation and average of tour lengths of ten runs, respectively; the column ‘Error’ shows the percentage of excess; the column ‘Time Avg.’ indicates the average running time in seconds.

5.2.3 Result for Concentric Tabu Search Algorithm (CTS)

In order to assess the efficacy of the proposed algorithm, fifteen traveling problems instances are deliberated. Evaluating the advantages of the algorithm and its performance we compared to the tabu search algorithm regarding to the tested problems. Initial solution is generated randomly; iteration size is 4000 for all experiments. The experiments illustrate that the proposed algorithm is more efficient in its ability of finding good solutions this is evident in its results. Table 4 and Table 5 below; summarize the results obtained for 15 problems available in the TSPLIB. The experiments were performed ten times for each problem. As can be seen from the outcomes presented below the propose algorithm quality is better compared to tabu search. Furthermore, the Ring Move algorithm is relatively better than the All Move algorithm as it shown in results.

Table 4: Result of the Concentric Tabu Search (Ring Move) Algorithm

#	Problem	Optimal	Best	Worst	STDEV	Error	Average	Time
1	KroA100	21282	27585	31717	1239	0.296	30191	14.9
2	kroA150	26524	38270	44442	1486	0.442	41304	28.9
3	kroA200	29368	46265	53987	2127	0.575	50483	47.9
4	kroB100	22141	28106	33102	1246	0.269	31001	24.8
5	kroB150	26130	37853	45283	2394	0.448	41373	28.3
6	kroB200	29437	46679	54573	2161	0.585	51283	84.8
7	kroC100	20749	27250	32711	1521	0.313	29728	14.1
8	kroD100	21294	27087	32197	1353	0.272	30495	21.8
9	kroE100	22068	28991	31219	706	0.313	30065	24.9
10	Bays29	2020	2152	2387	80	0.065	2304	2.61
11	Berlin52	7542	8034	9440	420	0.065	8860	5.62
12	Eil101	629	715	808	29	0.136	752	13.1
13	Lin105	14379	20170	24536	1128	0.402	21840	14.2
14	Ch150	6528	9397	10054	203	0.439	9801	27.6
15	Rat195	2323	3504	3924	142	0.508	3697	44.3

Table 4 prove that the suggested algorithm is more efficacious compared to the tabu search algorithm. As we can see the results of the proposed algorithm are better.

Table 5: Result of the Concentric Tabu Search (All Move) Algorithm

#	Problem	Optimal	Best	Worst	STDEV	Error	Average	Time
1	KroA100	21282	28491	31963	1150	0.3387	30409	15.38
2	kroA150	26524	38604	45761	1978	0.4554	41922	29.61
3	kroA200	29368	50915	57566	2277	0.7336	54754	25.33
4	kroB100	22141	27833	32938	1560	0.2570	30421	15.29
5	kroB150	26130	38865	44606	1918	0.4873	42010	29.73
6	kroB200	29437	49200	53712	1298	0.6713	51278	24.72
7	kroC100	20749	27678	31880	1546	0.3339	29916	15.09
8	kroD100	21294	28802	30328	569	0.3525	29615	15.47
9	kroE100	22068	27673	32122	1216	0.2539	30503	15.36
10	Bays29	2020	2036	2102	21	0.0079	2072	1.667
11	Berlin52	7542	8026	9195	331	0.0641	8547	3.302
12	Eil101	629	754	828	22	0.1987	784	9.042
13	Lin105	14379	20402	26282	1671	0.4188	21570	8.615
14	Ch150	6528	9007	10045	296	0.3797	9742	16.15
15	Rat195	2323	3507	3952	149	0.5096	3707	24.18

Table 5 gives the results of fifteen symmetric problems of size from 29 to 200. All problems have better results compared to the tabu search algorithm. Only the results of problems: kroB100 of size 100, kroE100 of size 100, Berlin52 of size 52, Bays29 of size 29, and Ch150 of size 150 could be solved better than the previous version of Concentric tabu search (Ring Move) algorithm as well as tabu search algorithm. Considering the result, outcome of the experiment is sensitive to the total of cities in the problem and total of iterations. Figure 15 illustrates how problems by using proposed algorithm and tabu search are converging to the optimal solution for the tested instances.

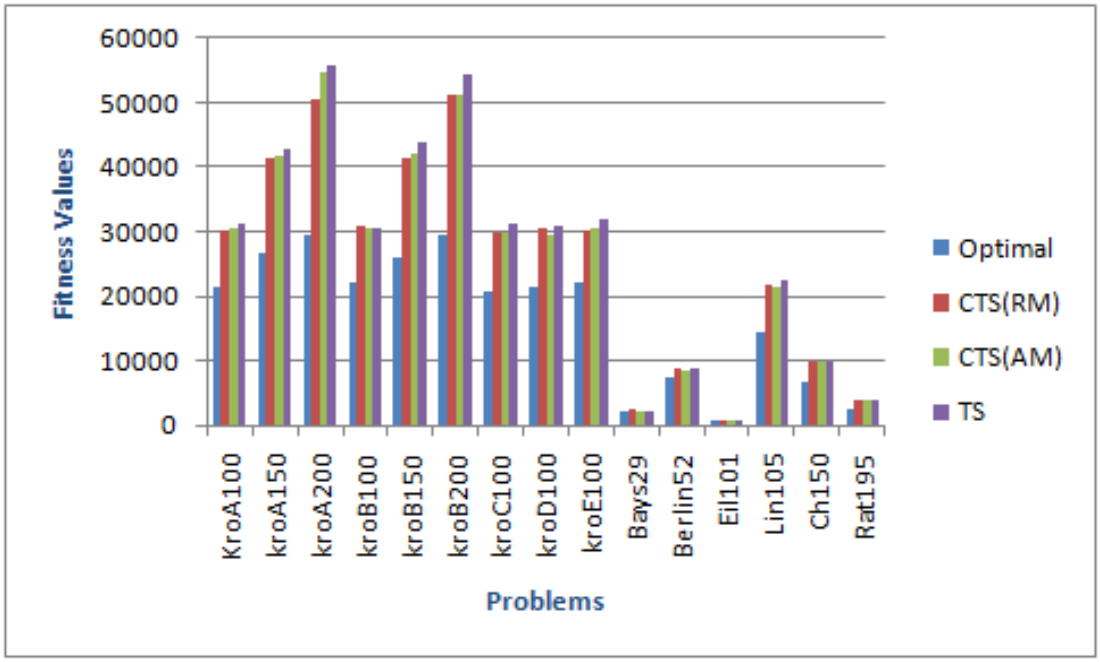

Figure 17: Results of TS, CTS (RM), and CTS (AM) for TSP

5.2.4 Result for Genetic Concentric Tabu Search Algorithm (GCTS)

In order to assess the proposed algorithm (GCTS), again the previous fifteen examples are considered. Control parameters are same as the parameters applied in traditional genetic algorithm and Concentric tabu algorithm. Local sarch algorithm applied only each fifty iteration to evolutionary process to keep the efficiency use of the hybrid algorithm and to reduce the computation time spent by local search algorithm. Ten runs were carried out to check the results obtained by GCTS. The result of the proposed algorithm was measured against the benchmark optimal solution, it can be noticed below in table 6 and in figure 16 that GCTS obtained better solution compaired to the other algorithms. Reported results show the effect of Concentric tabu local search algorithm with the genetic as a Metaheuristic algorithm. By combining the two, reaching the optimal or near-optimal solution became higher in the instances of the TSP problems considered.

Table 6: Result of the Genetic Concentric Tabu Search Algorithm

#	Problem	Optimal	GCTS (RM)	Time	GCTS(AM)	Time
1	KroA100	21282	21727	85.83	22096	72.95
2	kroA150	26524	28100	118.2	28089	101.6
3	kroA200	29368	31217	296.1	31371	278.9
4	kroB100	22141	22236	72.15	22179	52.15
5	kroB150	26130	27418	147.6	27496	137.6
6	kroB200	29437	31463	276	31679	177.7
7	kroC100	20749	21159	42.86	21203	73.27
8	kroD100	21294	21934	72.83	22349	62.11
9	kroE100	22068	22913	32.8	22327	27.92
10	Bays29	2020	2020	11.3	2020	10.24
11	Berlin52	7542	7542	26.6	7542	32.24
12	Eil101	629	632	51.58	635	35.72
13	Lin105	14379	14828	27.23	14818	26.97
14	Ch150	6528	6879	67.76	6893	33.90
15	Rat195	2323	2568	118.7	2564	97.38

Table 6 gives the results obtained by executing the GCTS algorithm on fifteen symmetric instances with cities between 29 to 200. It's worth mentioning that all problems have optimal or nearest to optimal solution. Instances: Bays29 and Berlin52 could be solved completely. Third column show the result of applying the first version (Ring Move) of Concentric tabu while the fifth column illustrate the results of applying the second version (All Move) of Concentric tabu search algorithm. The column "Time Avg." refers to the average running time in minutes. Figure 16 shows how the problems by using hybrid algorithm are converging to the optimal solution.

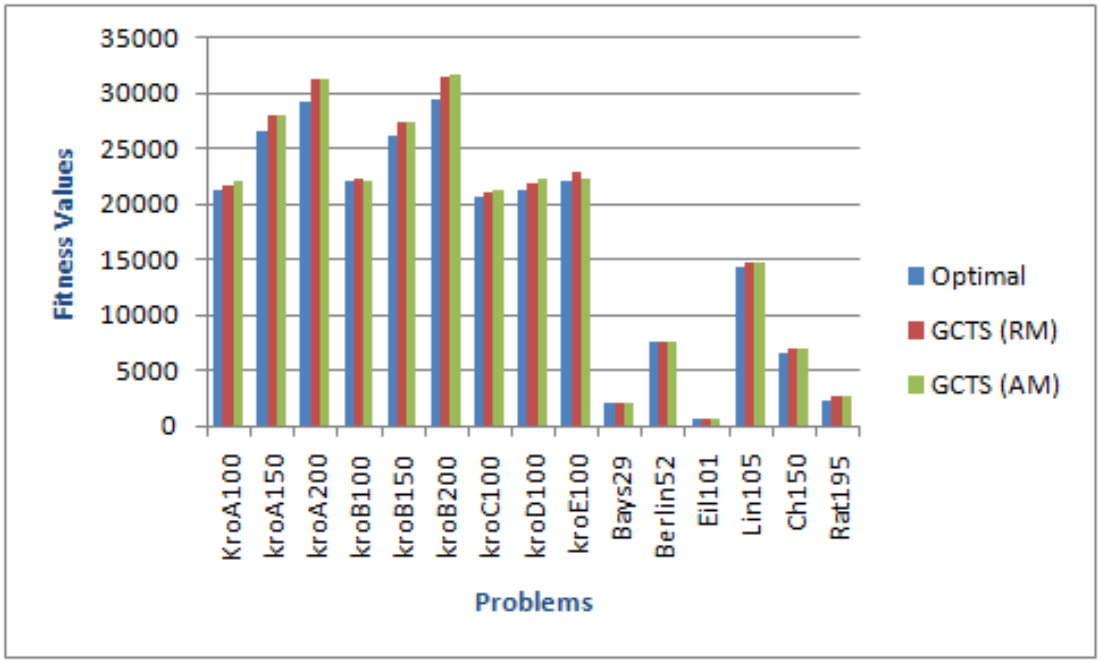

Figure 18: Results of the GCTS (RM) and GCTS (AM) for TSP

Chapter 6

CONCLUSION

In this research, Concentric Tabu Search Algorithm (CTS) has been proposed for work out the TSP as a cornerstone for heuristics designed for combinatorial optimization. Similarly, the CTS algorithm was compared to the traditional tabu search algorithm wherein the performance of CTS indicates its superiority over the traditional tabu search techniques. To enhance performance, CTS was combined with genetic algorithm in order to produce higher quality solutions. Consequentially, it can deduced that local search techniques cooperate with global search techniques to enhance the search space better in order to find more efficient solutions, considering that number of iterations have an effective role in finding optimal solution.

Finally, future work should try to enhance the efficiency of the proposed algorithm in minimizing time, especially for GCTS. In addition to this, further tests of the algorithm on more convoluted problems are needed to give a more accurate estimation of characteristic of the proposed algorithm.

REFERENCES

[1] Zvi, D. (2005). The extended concentric tabu for the quadratic assignment problem. *European Journal of Operational Research*, 160, 416-422.

[2] Zvi, D. (2008). Tabu Search and Hybrid Genetic Algorithms for Quadratic Assignment Problems. *European Journal of Operational Research*, 35, 90-107.

[3] Wassim, J. (2008). Local Search Techniques: Focus on Tabu Search.

[4] Sumanta, B. (2012). Tabu Search Implementation on Traveling Salesman Problem and Its Variations: A Literature Survey. *American Journal of Operations Research*, 2, 163-173.

[5] Christopher, M., & Gary, G. (2004), A Hybrid Evolutionary Algorithm for Traveling Salesman Problem. *International Conference on Evolutionary Computation*, 2, 1473-1478.

[6] Shubhra, S., Sanghamitra, B., & Sankar, K. (2004). New Operators of Genetic Algorithms for Traveling Salesman Problem. International Conference on Pattern Recognition, 2, 497-500.

[7] Chetan, C., Shah, S., & Mahesh, P. (2011). Comparison of Parents Selection Methods of Genetic Algorithm for TSP. *International Conference on Computer Communication and Networks*, 1, 102-105.

[8] Zvi, D. (2002). A New Heuristic for the Quadratic Assignment Problem. *Journal of Applied Mathematics and Decision Sciences*, 6, 163-173.

[9] Saloni, G., & Poonam, P. (2013). Solving travelling Salesman Problem Using Genetic Algorithm. International Journal of Advanced Research in Computer Science and Software Engineering, 3, 376-380.

[10] Fred, G., James, K., & Manuel, L. (1995). Genetic Algorithms and Tabu Search: Hybrids for optimization. *American Journal of Operations Research*, 22, 111-134.

[11] Bajeh, A. & Abolarinwa, K. (2011). Optimization: A Comparative Study of Genetic and Tabu Search Algorithms. *International Journal of Computer Applications*, 31, 43-48.

[12] Guohui, Z., Liang, G., & Yang, S. (2010). A genetic Algorithm and Tabu Search for Multi Objective Flexible Job Shop Scheduling Problems. *International Conference on Computing, Control and Industrial Engineering*, 1, 251-254.

[13] David, L., Robert, E., Vasek, C., & William J. (2006). The Traveling Salesman Problem: A Computational Study. Princeton University Press.

[14] Arananayakgi, A. (2014). Reduce Total Distance and Time Using Genetic Algorithm. *International Journal of Computer Science and Engineering Technology*, 5, 815-819.

[15] Otman, A., & Jaafar, A. (2010). A comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem. *International Journal of Computer Applications*, 31, 49-57.

[16] Alexandar, S. (2005). On the history of Combinatorial Optimization (Till 1960). *American Journal of Mathematics*, 12, 1-68.

[17] Zar, H., May, K. (2011). An Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem. *International Conference on Information Communication and Management*, 16, 54-59.

[18] Thamilselvan, R., & Balasubramanie, P. (2009). Integrating Genetic Algorithm, Tabu Search Approach for Job Shop Scheduling, *International Journal of Computer Science and Information Security*, 2, 42-53.

[19] Fiechter, C. (1994). A Parallel Tabu Search Algorithm for Large Travelling Salesman Problem. *International Journal of Discrete Applied Mathematics*, 51, 243-267.

[20] Lionardo, Z. (2006). The Traveling Salesman Problem: A Comprehensive Survey. *European Journal of Operational Research*, 59, 231-247.

[21] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html.