Project/Thesis No.:

MODIFIED SPIDER MONKEY OPTIMIZATION FOR TRAVELING SALESMAN PROBLEM

BY

SAFIAL ISLAM AYON

Roll: 1407041

A thesis submitted to the Department of Computer Science & Engineering in partial fulfilment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

KHULNA 9203, BANGLADESH

February, 2019

MODIFIED SPIDER MONKEY OPTIMIZATION FOR TRAVELING SALESMAN PROBLEM

BY

SAFIAL ISLAM AYON

Roll: 1407041

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY
KHULNA 9203, BANGLADESH

February, 2019

CERTIFICATION

The thesis titled "**Modified Spider Monkey Optimization for Traveling Salesman Problem**" submitted by Safial Islam Ayon, Roll No: 1407041, Academic Year: 2017-18, for partial fulfillment of the requirements for the degree of "Bachelor of Science in Computer Science and Engineering".

Supervisor

Dr. Muhammad Aminul Haque Akhand

Professor

Department of Computer Science and Engineering

Khulna University of Engineering and Technology

ACKNOWLEDGEMENTS

First and foremost, I must sense grateful to and wish to acknowledge my insightful indebtedness to Dr. Muhammad Aminul Haque Akhand, Professor of Department of Computer Science and Engineering and the supervisor of the thesis. His unfathomable knowledge in the field of Swarm Intelligence influenced me to carry out this thesis up to this point. His endless endurance, Scholarly guidance, continual encouragement, constant and lively supervision, constructive criticism, priceless suggestion made it possible to come up to this phase. Without his inspiring, enthusiasm and encouragement, this work could not be completed.

Last, but by no means least, I thank Allah for the talents and abilities I was given that made it possible to undertake this thesis.

ABSTRACT

Traveling Salesman Problem (TSP) is the problem faced by a salesman who starts from a specific city and travels all the other cities in the shortest conceivable path. TSP is the most popular combinatorial optimization as many real-world problems can be formulated as an instance of the TSP. Nearly every year, new methods for solving various problems (e.g., engineering, optimization) are verified on the TSP considering it as a standard test bench. Recently, Spider Monkey Optimization (SMO) is developed for numerical optimization inspired by the intelligent foraging behavior of spider monkeys. This thesis investigates a new effective optimization method to solve TSP based on the social behavior of spider monkeys. The proposed Modified SMO (MSMO) tackles the challenges of developing a novel method for TSP updating features of SMO as well as introducing the required features. In MSMO, every spider monkey represents a TSP solution; and Swap Sequence (SS) and Swap Operator (SO) based operations are employed for interaction among monkeys to obtain the optimal TSP solution. The SOs are generated using the experience of a specific spider monkey as well as the experience of other members (local leader, global leader, or randomly selected spider monkey) of the group. The proposed method has been examined on a large number of benchmark TSPs and outcomes are compared to other prominent methods. Experimental results demonstrate the effectiveness of the proposed MSMO to solve TSP.

ABBREVIATIONS

SMO	Spider Monkey Optimization
MSMO	Modified Spide Monkey Optimization
SM	Spider Monkey
SS	Swap Sequence
SOs	Swap Operators
TSP	Traveling Salesman Problem
SI	Swarm Intelligence
FFSS	Fission-Fusion Social Structure
LLP	Local Leader Phase
GLP	Global Leader Phase
LLLP	Local Leader Learning Phase
GLLP	Global Leader Learning Phase
LLDP	Local Leader Decision Phase
GLDP	Global Leader Decision Phase
MG	Maximum Group
BSS	Basic Swap Sequence
PS	Partial Search
PSO	Particle Swarm Optimization
ACO	Ant Colony Optimization
VTPSO	Velocity Tentative Particle Swarm Optimization
SD	Standard Deviation
STPSO	Self-Tentative Particle Swarm Optimization
SSPSO	Swap Sequence-based Particle Swarm Optimization

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Spider Monkey Optimization (SMO) is a recently developed population-based optimization technique on the metaphor of social behavior of spider monkeys. It particularly finds its place in the category of swarm intelligent techniques and it also becomes popular to solve many difficult optimization problems. Each spider monkey updates its position in the search space from time to time according to the local leader experience and global leader experience. SMO has been proven to succeed in continuous problems (e.g. function optimization) as it was proposed for such problems and much work has been done effectively in this area. SMO has also found as an efficient method to solve many combinatorial problems. In this thesis, given attention to solve Traveling Salesman Problem (TSP) with existing SMO based methods and investigate Partial Search (PS) mechanism in the SMO update operation and finally modify the SMO algorithm such that using this algorithm the TSP can easily be solved.

1.1 Background

Swarm intelligence (SI) is a meta-heuristic method in the field of nature-inspired procedures that is used to solve optimization problems [1]. It is based on the social behavior of animals [2]. SMO is a recently developed nature-inspired stochastic optimization technique, which particularly finds its place in the category of SI. SMO is inspired by the foraging behavior of spider monkeys. The social behavior of spider monkeys is an example of a fission-fusion social structure (FFSS). The spider monkey lives in a large group and shows intelligence foraging behavior. They search for food in a different direction and share their information with each other. Their intelligence technique to search for food is the source of motivation to develop the algorithm. When animals are living together and share their information with each other the group members are benefited.

Like other metaheuristics algorithm, it starts with uniformly generated random initial positions of all spider monkeys in the swarm. This position is updated with iterations. In a group of spider monkey when the members are failed to search for food, the group leader divides the group into some subgroups. The subgroups also have a leader. All the group's leaders are female. All the members search for food and communicate with each other via emitting voice. If the subgroups also failed to search for food the subgroup leader divide the group again [4]. When the number of groups is reached a maximum number then the main group leader combines all the subgroups into one large group.

SMO has been proven to succeed in continuous problems (e.g. function optimization) as it was proposed for such problems and much work has been done effectively in this area. In this thesis, using the Modified SMO (MSMO) algorithm trying to solve combinatorial problems such as Traveling Salesman Problem (TSP). TSP is a well-studied combinatorial optimization problem in which a salesman is required to complete a tour with minimum distance visiting all the assigned cities exactly for once. To solve TSP with SMO, a different consideration for the spider monkeys' position is required. Each spider monkey represents a complete tour to solve TSP and update the position of a spider monkey give a new tour.

In recent years Swap sequence (SS) are a popular method to solve TSP [3, 5]. A Swap Sequence (SS) is a collection of several Swap Operators (SOs) and may define as SS = (SO_1, SO_2, SO_3, …, SO_n), where SO denotes Swap Operator [3, 5, 8]. A SO indicates two positions in the tour that might be swapped. All SOs of a SS are applied to maintain order on a particle and gives a new tour, i.e. a spider monkey having a new solution in the SMO.

In PSO there are several proposed methods where TSP is solved using SS. The SS based PSO (SSPSO) [5], Self-Tentative PSO (STPSO) and Enhance Self Tentative PSO (ESTPSO) [7], Velocity-Tentative PSO (VTPSO) [15], are some of the methods where SS are used to solve TSP.

1.2 Objectives of the Thesis

MSMO algorithm is completely a new technique to solve TSP. Here the both Swap Sequence (SS) and Partial Search (PS) are used. In SS, the new tour of TSP is considered after applying all the SOs of a SS and no intermediate measure is considered. But in PS, all the SOs are applying to the solution one by one sequentially. So, there might be a chance to get a better tour with some of the SOs instead of all the SOs. The Objective of the study is to solve TSP using the SMO algorithm with the help of SS.

The study will be carried out with the following specific objectives:

- Review other methods to solve TSP.
- Apply MSMO algorithm in TSP.
- Investigate SS and PS mechanism in MSMO and develop the algorithm that might be an efficient method to solve TSP.
- Compare the performance of proposed MSMO for TSP with exiting SS based other methods in solving benchmark TSPs.

1.3 Organization of the Thesis

The main attraction of this thesis is to present a new MSMO based algorithm to solve TSP. The thesis has five chapters. An introduction to SMO has been given in Chapter 1. Chapter wise overviews of rest of the thesis are as follows: Chapter 2 is for literature review that includes a brief description of standard SMO and previous related work to SMO. Chapter 3 explains the proposed Modified Spider Monkey Optimization (MSMO) to solve TSP in details. Chapter 4 reports the experimental result of Modified Spider Monkey Optimization (MSMO). Also, in this chapter compares the performance of the proposed MSMO with existing methods to solve benchmark TSPs. Chapter 5 is for the conclusions of this thesis together with the outline of future directions of research opened by this work.

CHAPTER 2

Literature Review

Spider monkey is a South American species of monkey which lives in a large group and shows intelligence in social behavior and foraging [56-57]. Spider monkeys search for food in a group under a group leader and share their information with each other. When spider monkeys fail to search for food, the group leader divides the group into some subgroups. The subgroups also have individual leaders and search for food in different directions. A subgroup leader may divide the group again if fail to search food [4]. Once upon a time, the main group leader combines all the subgroups into the large main group. The social behavior of spider monkeys is an example of a Fission-Fusion Social Structure [4] where group members are benefited from living together and sharing information. Inspired by the intelligent behavior of spider monkeys, Bansal et al. [1] proposed the Spider Monkey Optimization (SMO) algorithm for numerical optimization. Like other metaheuristics algorithms, SMO starts with a uniformly generated random initial positions of all spider monkeys in the swarm. The positions of individual monkeys are updated over iterations through interacting with others. The SMO algorithm also determines how to update the position of a spider monkey. Each spider monkey updates its position based on current position, local leader position, global leader position, and randomly selected spider monkey position. In this chapter, we will discuss basic SMO algorithm and some features of SMO algorithm.

2.1 Standard Spider Monkey Optimization (SMO)

The SMO algorithm is an example of FFSS based social behavior of spider monkey. The searching mechanism is employed in SMO for numeral optimization. Like other SI algorithms, each monkey of SMO represents a solution having a position in a multi-dimensional search space. SMO starts with a population of uniformly generated random initial positions of the monkeys in the swarm. The positions are updated in every iteration through interactions among the monkeys. The best position of all the monkeys in the search space is called the global leader. The best solution for each group is known as the local leader of that group. Figure 2.1 indicates the global leader and local leader

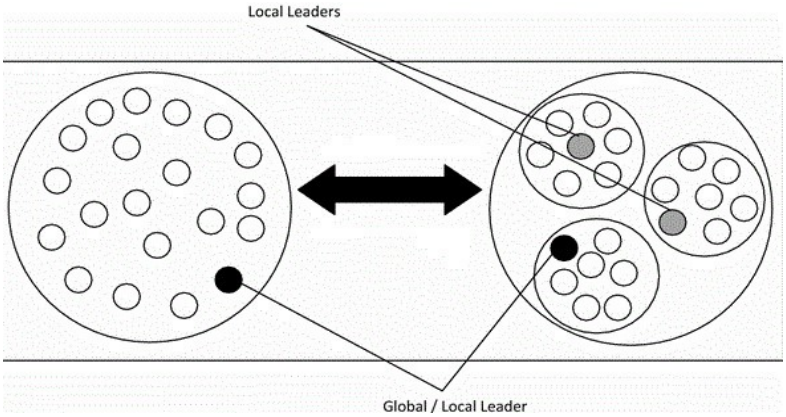

Figure 2.1: Global leader and Local leader in a spider monkey group.

in a spider monkey group.

The foraging behavior of spider monkey is divided into four phases.

1) They start searching for food and evaluates their distance from the food.

2) They update their positions according to the distance from the food and evaluate the distance from the food again.

3) The local leader updates its best position in his group and if the position is not updated a predetermined time then the group members start again searching food in different directions.

4) The global leader also updates its best position and if the group size exceeds the maximum number then the global leader combines all the group into one single group or else its divide the group into some subgroups. Here two important control parameters: GlobalLeaderLimit and LocalLeaderLimit, which helps local leader and global leader to take the right decisions.

LocalLeaderLimit is used when the local leader of a group is not updated a fixed number of times. The fixed number of times is mentioned as LocalLeaderLimit. In this situation, the group members search for food in different directions. The GlobalLeaderLimit is used for similar reasons for global leader. The global leader divided the group into small groups or combine all the groups into one single group according to the value of GlobalLeaderLimit.

The algorithm follows self-organization as well as separation of labor properties for

finding intelligent swarming activities of the animal. It demonstrations positive feedback mechanisms of self-organization because, here a spider monkey updates its position using local leader, global leader, and self-experience. When the global leader divided the groups into subgroups, it represents the divisions of labor property. Local leader limit and Global leader limit provides negative feedback to help local and global leaders for their decisions.

The algorithm is inspired by the behavior of spider monkey, but it is different from the natural foraging behavior of spider monkey. Spider monkeys use a different type of communication technique which is not possible to simulate. In this way, the proposed strategy is different from the real forging behavior of spider monkeys.

Similar to other population-based algorithms, SMO is a trial and error based collaborative iterative process. There are six phases in SMO namely Local Leader phase, Global Leader phase, Local Leader Learning phase, Global Leader Learning phase, Local Leader Decision phase, and Global Leader Decision phase and each one has a different purpose. A brief description of each phase is given below and the notations used in the algorithm are provided in Table 2.1.

Table 2.1: Notations used in the SMO algorithm

SM_{ij}	*ith* spider monkey in *jth* dimension.
R (p, q)	Uniform random number between *p* and *q*.
SM_{new}	New position of a spider monkey.
pr	Perturbation rate.
$fitness_i$	Fitness of the *ith* spider monkey.
Maxfitness	Maximum fitness of the swarm.
$probability_i$	Probability of selection of *ith* member in the swarm.
GL_j	Global leader in *jth* dimension.
RSM_{rj}	Random spider monkey in the *jth* dimension.
LL_{kj}	Local leader in the *jth* dimension and in the *kth* group.
$SMmin_j$	Minimum bounds of *jth* direction.
$SMmax_j$	Maximum bounds of *jth* direction.
max_fitness	Maximum fitness of the *kth* group.
fitness(i)	Fitness of the ith Spider Monkey.

2.1.1 Initialization of Population

Originally a population comprised of N spider monkeys signifying a D-dimensional range SMi where $i=1, 2, ..., N$ and i represents ith spider monkey. Each spider monkey (SM) exhibits possible results of the problem under consider. Each SMi is initialized as below:

$$SM_{ij} = SMmin_j + R\ (0,\ 1) \times (SMmax_j - SMmin_j) \tag{2.1}$$

Here, $SMminj$ and $SMmaxj$ are limits of SM_i in jth vector and $R(0,1)$ is a random number between 0 to 1.

2.1.2 Local Leader phase (LLP)

In this phase, every spider monkey is updated using his previous experience, the local leader experience, and a randomly selected spider monkey of that group. It compares fitness new location and current location and applies greedy selection. Depend on the perturbation rate the spider monkey update their position. The equation to update new position is given below:

$$SMnew_{ij} = SM_{ij} + R(0,1) * (LL_{KJ} - SM_{IJ}) + R(-1,1) * (SM_{RJ} - LL_{IJ}) \tag{2.2}$$

Here the jth dimension of the $ith\ SM$ is denoted by SM_{ij}, the jth dimension of the kth local group leader position is denoted by LL_{KJ}. SM_{RJ} indicates the jth dimension of the $rth\ SM$ which is selected randomly from the kth group such that $r = i$, $R(-1,1)$ is a uniformly distributed random number between -1 and 1.

2.1.3 Global Leader phase (GLP)

Like local leader phase, depending upon the observation of global leader and mates of local troop, spider monkey updates their location. The position upgrade equation for Global leader phase is as follow:

$$SMnew_{ij} = SM_{ij} + R(0,1) * (GL_J - SM_{IJ}) + R(-1,1) * (SM_{RJ} - LL_{IJ}) \tag{2.3}$$

Here GLj is the jth dimension of the global leader position and $j = [1, 2, ..., D]$ is an arbitrarily selected index. The spider monkey updates their position based on the probability value $prob_i$. If the probability value is greater than a randomly selected number between 0 to 1 then only the spider monkey updates its position using its current position, global leader position and randomly selected spider monkey. The formula for

calculating the probability of ith spider monkey is given below:

$$prob(i) = 0.9 * \frac{Fitness(i)}{max\_fitness} + 0.1 \quad (2.4)$$

Here *fitness(i)* means the fitness value of the *ith SM* and *max_fitness* signifies the maximum fitness in the group. The fitness of the recently generated position of the SMs is assessed and the comparison is done with the old one and the better position is adopted.

2.1.4 Global Leader Learning (GLL) phase

Here greedy selection strategy is applied to the population which modifies the position of global leader i.e. the location of *SM* which has the best fitness in the group is chosen as the modified global leader location. Also, it is verified that the global leader location is modifying or not and if the global leader does not update then the *Global Limit Count* is incremented by *1*.

2.1.5 Local Leader Learning (LLL) phase

Like global leader learning phase, local leader position is modified by implement greedy selection in that population i.e. the location of *SM* which has the best fitness among the entire group is chosen as the new location of local leader. If the position of the local leader of a spider monkey of a particular group does not get updated, *Local Limit Count* is increment by *1*.

2.1.6 Local Leader Decision (LLD) phase

Here, updating of local leader location is done in two ways.

1) arbitrary initialization
2) mixing information obtained via global and local leader.

If local leader location is not modified up to a precalculated limit named as LocalLeaderLimit through an equation based on perturbation rate:

$$SMnew_{ij} = SM_{ij} + R(0,1) * (GL_J - SM_{IJ}) + R(0,1) * (SM_{IJ} - LL_{KJ}) \quad (2.5)$$

Here, in the equation, we saw that modified dimension of a particular spider monkey (*SM*) is fascinated towards global leader and oppose local leader. Moreover, modified SM's fitness is determined.

2.1.7 Global Leader Decision (GLD) phase

In this phase, global leader location is examined and if the modification is not done up to predetermine time named as GlobalLeaderLimit then group leader divides the group into a small group. Primarily population division of that group is done into two classes and further three, four and so on until the upper bound called groups of maximum number is reached. Meanwhile, local leaders are selected using local leader learning phase method for newly formed subclasses.

2.2 Pseudocode of Spider Monkey Optimization

There are six phases in SMO namely Local Leader phase, Global Leader phase, Local Leader Learning phase, Global Leader Learning phase, Local Leader Decision phase, and Global Leader Decision phase. Figure 2.2 shows the steps of Spider Monkey Optimization (SMO). Local Leader phase is responsible for producing a newly updated position for every spider monkey in the group (Step 3.a). In the Global Leader phase, a probability is calculated (Step 3.b) first then the position of all the spider monkeys are updated (Step 3.c) If the position is better than the previous position, the position is updated to the new position otherwise it retains the current solution. Local Leader and Global Leader are selected in the Local Leader Learning phase and Global Leader Learning phase respectively (Step 3.d). In the Local Leader Decision phase and Global Leader Decision phase, the Local Leaders initialize all the members of the groups again (Step 3.e) and the Global Leader takes the decision to divide the group into subgroups or combine (i.e. fuse) all the groups into one group (Step 3.f).

SMO follows self-organization as well as separation of labor properties for finding intelligent swarming activities of the animals. It demonstrates positive feedback mechanisms of self-organization because a spider monkey updates its position using local leader, global leader, and self-experience. When the global leader divides the group into subgroups, it represents the division of labor property observed in many species. SMO considers two important control parameters: Global Leader Limit and Local Leader Limit, which help local leader and global leader to take the right decision.

Step 1: Initialization

Step 2: Select **Local Leader** and **Global Leader**

Step 3: For each Spider Monkey

a. Generate new position
b. Calculate the probability
c. Based on the probability update position again
d. Select new **Local Leader** and **Global Leader**
e. Redirect all the members of the group if **Local Leader** is not updating for a specific time.
f. Splits the group into subgroups or combine all the groups into one group if **Global Leader** is not updating for a specific time.

Step 4: Goto **Step 3** if the termination condition is not met

Step 5: Return the **Global Leader** as the final solution

Figure 2.2: Standard SMO algorithm.

Finally, the global leader will provide the final outcome (i.e. solution) of SMO.

In figure 2.3 the flowchart of the spider monkey optimization is shown.

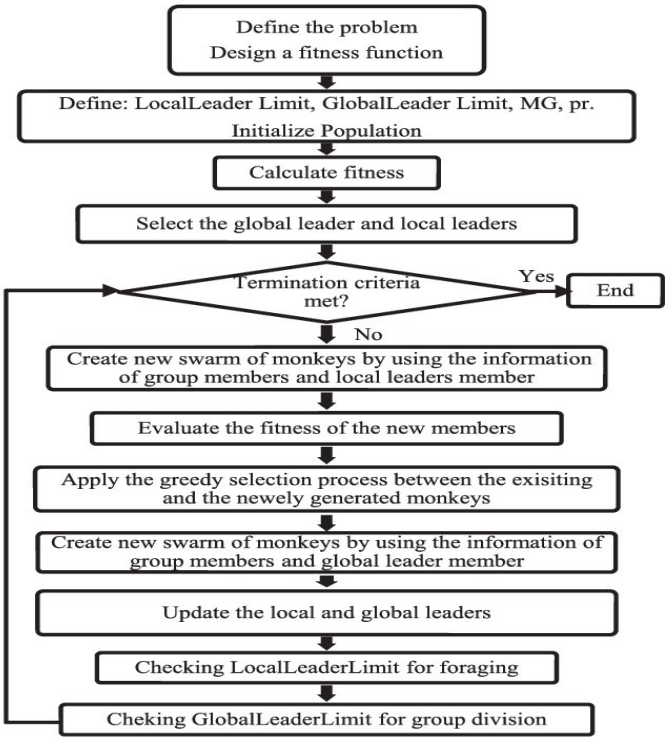

Figure 2.3: Flowchart of Standard SMO algorithm.

2.3 Features of SMO

SMO has found a good number of applications in many domains within a very short period of time [58]. SMO has been applied to automatic generation of control [41], designing electromagnetic antenna arrays [59], rule mining for diabetes classifications [60], designing optimal fuzzy rule-base for a Tagaki-Sugeno-Kang (TSK) fuzzy control system [36], improving quality and diversity of particles and distribute them in particle filters for providing a robust object tracking framework [30], CDMA multiuser detection [61], solving optimal reactive power dispatch problem [22], optical power flow, pattern synthesis of sparse linear array and antenna arrays [23], routing protocol wireless sensor networks [24], dynamic stability improvement of VSC-HVDC connected multi machine power system [26], modified spider monkey optimization algorithm [27-28], optimization in electromagnetics [29], antenna optimization [31], estimation frequency-modulated sound waves [37], Image compression [34], fuzzy logic [38], numerical classification [35], optimizing frequency in microgrid [62], economic dispatch problem [63], optimizing models of multi-reservoir system [64], and energy efficient clustering for WSNs [33].

A number of studies on modified SMO for optimization problems in different domains have also been reported in the literature. Some of the SMO algorithms are modified to solve global optimization problems [65], to improve the local search (i.e., exploitation of the search space) capability of algorithm [66, 67] and proposed fitness-based position update strategy for the spider monkeys [21], modified ageist SMO incorporating age of the monkeys that further divides the groups of monkeys into subgroups according to age groups based on different levels of ability [25], binary SMO [23], modified SMO for constraint continuous optimization [32], and modified SMO incorporating Nelder–Mead method to enhance local search [65].

All these applications listed above are numeral optimization in a broader sense where mathematical operations (e.g., additional, subtraction, multiplication and division) are easy to implement for interaction within elements and individuals in SMO. New operators have to be designed or SMO has to be modified for new applications.

CHAPTER 3

Modified Spider Monkey Optimization to solve TSP

SMO conceived the basic features of spider monkeys for numerical optimization, it is completely different and more difficult to adapt for discrete optimization problems like TSP. The proposed modified version SMO of this study tackles the challenges of applying SMO to TSP by introducing a number of new features and operators to SMO. In original SMO, fitness of a solution is based on its objective function while SMO for TSP the fitness value is the tour cost of a particular solution. There are six phases to solve the SMO for TSP problem: Local Leader phase, Global Leader phase, Local Leader Learning phase, Global Leader Learning phase, Local Leader Decision phase, and Global Leader Decision phase. In the proposed modified SMO, every spider monkey represents a TSP solution; and Swap Sequence (SS) and Swap Operator (SO)-based operations are considered for interaction among monkeys to get the optimal TSP solution. An SS is a group of SOs and each contained a pair of indexes on the TSP tour. A new tour is generated transforming a TSP tour swapping city indicated by SOs of a SS. The SS generation to update a monkey for a better solution is the key point of the method and SS of a particular spider monkey is generated with interaction with other members of the group. In case of updating a solution of spider monkey with generated SS a Partial Search (PS) technique is considered to achieve the best outcome with full or partial SS. The proposed method has been tested on a suite of benchmark TSPs and is shown to outperform other prominent metaheuristic methods applied to TSP. In this chapter, the Traveling Salesman Problem (TSP), some existing methods to solve TSP, and proposed MSMO to solve TSP are discussed.

3.1 Traveling Salesman Problem (TSP)

The traveling salesman problem (TSP) is a well-known combinatorial optimization problem [12, 13]. TSP requires to find the shortest circular tour visiting every city exactly once from a set of given cities [14]. The TSP problem can be formally described as follows, suppose that there is one salesman who wants to visit N number of cities,

and his objective is to find out the shortest Hamiltonian cycle through which lie can visit all the cities once and only once, and finally returns to the starting city. More formally, given N cities. TSP requires a search for a permutation π: *{0, ..., N - 1)* → *{0, N – 1}*, using a cost matrix $C = [C_{ij}]$, where C_{ij} denotes the cost of the travel from city i to j, which minimizes the path length [13]:

$$f(\pi, C) = \sum_{i=0}^{N-1} C_{\pi(i),\ \pi((i+1) mod\ N)} \tag{3.1}$$

where $\pi(i)$ denotes the city at i^{th} location in the tour.

TSP can be modeled as an undirected weighted graph, such that cities are the graph's vertices path are the graph's edges, and a path's distance is the edge's length. It is a minimization problem starting and finishing at a specified vertex after having visited each other vertex exactly once. Often, the model is a complete graph *(i.e.* each pair of vertices is connected by an edge). If no path exists between two cities, adding an arbitrarily long edge will complete the graph without affecting the optimal tour. There are two types of TSP: *symmetric* and *asymmetric.* In the *symmetric* TSP, the distance between two cities is the same in each opposite direction, forming an undirected graph. This symmetry halves the number of possible solutions. In the *asymmetric* TSP, paths may not exist in both directions or the distances might be different, forming a directed graph.

Ideas related to TSP have been around for a long time. In 1736, Leonard Euler studied the problem of finding a round trip through seven bridges in Konigsberg. In 1832, a handbook was published for German traveling salesmen, which included examples of tours. In the 1850s, Sir William Rowan 1-lamilton studied Hamiltonian circuits in graphs. He also marketed his Icosian Game, based on finding tours in a graph with 20 vertices and 30 edges. In the early 1930s, Karl Menger discussed the problem with colleagues in Vienna and Harvard. In the late 1930s, the problem reappeared at Princeton University Hassler Whitney called it the TSP. Fig. 3.1 indicates an optimal tour for TSP with several typical cities marked in dot.

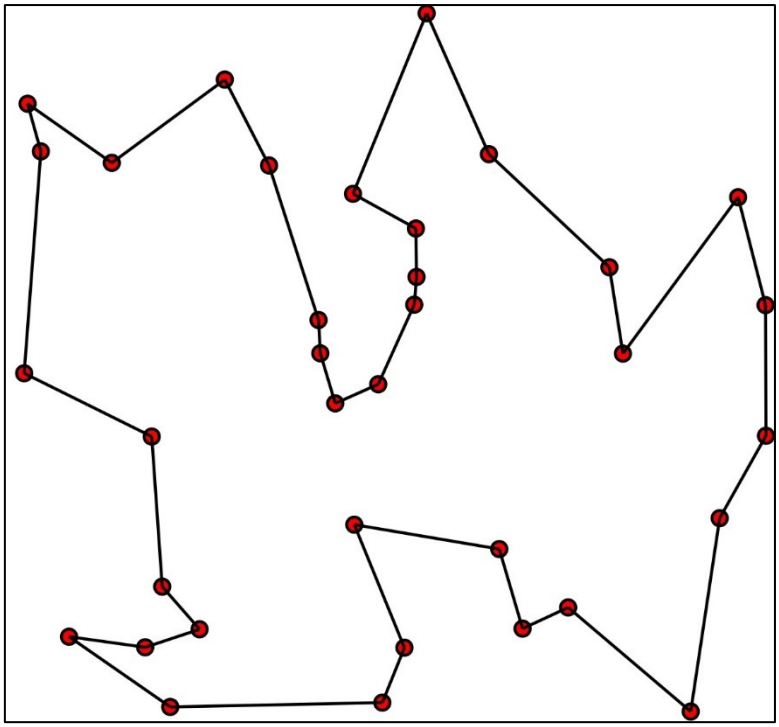

Figure 3.1: An optimal tour for a typical Traveling Salesman Problem (TSP).

TSP demonstrates all the parts of combinatorial optimization and comes under the set of NP-hard problems which cannot be solved optimally in polynomial time [19]. Solving TSPs is a significant part of applications in numerous practical problems within daily live [20]. There are several real-life applications of TSP such as: vehicle routing, computer wiring, X-Ray crystallography, DNS sequencing, DNS fragments, printed circuit board, order picking problem in warehouses [11]. In many applications, additional constraints such as limited resources or time windows make the problem considerably harder. TSP is the most popular combinatorial problem and interest grows in recent years to solve it new ways. Almost every new approach for solving engineering and optimization problems has been tested on the TSP [9, 10, 39, 40] as a general test bench.

3.2 Existing methods to solve TSP

TSP is a popular combinatorial problem and a number of methods [3,5,7,8,12,14,41-45] have been investigated for solving TSP up to date. There are several methods like: branch and bound [46], cutting plane [47], Nearest Neighbor [48], Insertion Heuristics [49], Greedy Heuristic [50], Hill Climbing [51], Simulated Annealing [52], Tabu

Search [53], Ant Colony Optimization (ACO) [54], and Genetic Algorithm [55] to solve TSP. But among those methods, we interested to solve TSP using Swap Sequence (SS) and Swap Operator (SO). Following subsections explain the operator SO and SS in detail and some methods where TSP is solved by using SS and SO.

3.2.1 Swap Operator (SO)

A Swap Sequence (*SS*) having several Swap Operators (*SOs*) is considered to transform a TSP solution into a new solution [5 – 7]. A TSP solution is a sequence of cities to be visited and a *SO* is a pair of position indexes which imply two cities to be interchanged (i.e., swapped) in the TSP solution. Suppose a TSP has five cities and a solution is $S = (1 – 4 – 5 – 3 – 2)$ and a SO is $SO(3,4)$, then the new solution S' is:

$$\begin{aligned} S' &= S + SO(3,4) \\ &= (1 – 4 – 5 – 3 – 2) + SO(3,4) \\ &= (1 – 4 – 3 – 5 – 2) \end{aligned}$$

Here '+' is not an arithmetic addition operation rather application of *SO* operation onto the solution. Swap Operator is an important factor to update TSP solution in this study and its operation is similar to mutation operation of GA.

3.2.2 Swap Sequence (SS)

A Swap Sequence [5-7] is a set of single or additional SOs that is applicable to a specific TSP solution in a sequence expressed by:

$$SS = (SO_1, SO_2, SO_3, \ldots, SOn) \tag{3.2}$$

Where $SO_1, SO_2, SO_3, \ldots, SO_n$ are *SOs*. A *SS* acts on a solution applying all its SO_s maintaining its sequence and then finally produces a new tour. This can be described by the following formula

$$\begin{aligned} S_2 &= S_1 + SS \\ &= S_1 + (SO_1, SO_2, SO_3, \ldots, SO_n) \end{aligned}$$

The order of SOs in the SS is important because the implementation of same SOs in a different order may give different solutions from the original solutions. This can be described by the following formula:

$$P' = P + SS$$
$$= P + (SO_1, SO_2, \ldots, SOn)$$
$$= (((P, SO_1), SO_2), \ldots, SO_n) \quad (3.3)$$

The SS may also get from solutions S_1 and S_2 in the following equation.

$$SS = S_2 - S_1$$
$$= (SO_1, SO_2, SO_3, \ldots, SOn) \quad (3.4)$$

Where '-' means need to apply SO_s of SS on solution S_1 to S_2. Suppose two solutions is $S_1 = (1 - 2 - 3 - 4 - 5)$ and $S_2 = (2 - 3 - 1 - 5 - 4)$ then $SS = SO(1,2), SO(2,3), SO(4,5)$.

Moreover, several SSs may be merged into a new SS, the operator ⁕ defines as merging operation. If $SS_1 = SO(1,2), SO(5,2)$ and $SS_2 = SO(5,3), SO(4,1)$ then new Swap Sequence *SS(new)* merging SS1 and SS2 is

$$SS(new) = SS_1 ⁕ SS_2$$
$$= \{SO(1,2), SO(5,2)\} ⁕ \{SO(5,3), SO(4,1)\}$$
$$= SO(1,2), SO(5,2), SO(5,3), SO(4,1)$$

3.2.3 Basic Swap Sequence (BSS)

It is also to be noted that various *SSs* applied to the same TSP solution might generate the same result. This type of *SSs* is called the equiponderant set of *SSs*. Among all this *SSs*, the *SS* which hold the minimum *SOs* is termed as the Basic Swap Sequence (*BSS*).

Suppose two *SSs* are: $SS_1 = \{SO(2,3), SO(3,2), SO(4,1), SO(5,3), SO(3,4)\}$ and $SS_2 = \{SO(4,1), SO(5,3), SO(3,4)\}$, which are applied to $S_1 = (5 - 1 - 2 - 3 - 4)$ independently, giving the outcome $S_2 = (3 - 1 - 5 - 4 - 2)$. Consequently, SS_2 is the *BSS* which also be found employing $S_2 - S_1$ using Eq. (3.4).

3.2.4 Partial Search (PS)

Partial Search (PS) based SS operations are found efficient in solving TSP [3, 15]. Since each and every SO implication gives individual TSP solutions, PS technique considers each one as tentative tour and returns the best one as the outcome. Suppose a solution is $S = (1 – 3 – 2 – 5 – 4)$ and $SS = \{SO(2,3), SO(1,2), SO(4,5)\}$. Using the three *SOs* the three tentative tours (T_1, T_2, and T_3) and their corresponding TSP costs may as follow:

Table 3.1: Apply Swap Sequence in a Solution

Applying Swap Sequence	Tentative TSP Tours	TSP Tour Cost
$S + SO(2,3)$	$T_1 = (1 – 2 – 3 – 5 – 4)$	$Cost_1 = f(T_1)$
$T_1 + SO(1,2)$	$T_2 = (2 – 1 – 3 – 5 – 4)$	$Cost_2 = f(T_2)$
$T_2 + SO(4,5)$	$T_3 = (2 – 1 – 3 – 4 – 5)$	$Cost_3 = f(T_3)$

All three tentative tours are feasible TSP solutions and an intermediate tentative solution can be better than the last one. Thus, *PS* return the solution having the lowest TSP cost [3]. In this study, *PS* technique is considered to achieve the best outcome with full or partial *SS* while updating a solution of spider monkey with generated *SS*.

3.2.5 Methods to solve TSP using Swap Sequence

Here several methods are discussed where TSP is solved by using Swap Sequence (*SS*). Some of the methods are discussed in the following subsections.

3.2.5.1 Swap Sequence-based PSO (SSPSO)

To solve TSP with PSO, the pioneer method [5] considered Swap Sequence (*SS*) as the velocity: we hereafter call the method as Swap Sequence based PSO (SSPSO) for TSP. In SSPSO, each particle represents a complete tour and conceives SS as a velocity of it. The algorithm gives a new tour and ultimately gives a complete tour after applying all the *SOs* in a *SS*. Each particle changes position to new tour solution based on its *SS* which consists of a set of *SOs* that depends on its previous best position and the best one among all the particles in the population. Figure 3.2 shows the steps of SSPSO algorithm.

Step 1: Initialization: define number of particles, termination criterion, tour cost. Assign a random tour and random Swap Sequence as velocity to each of the particle. Consider previous best tour as current tour and global best tour as the best one among them.

Step 2: For each particle in the swarm

a) Calculate velocity.

b) Update solution.

c) Update particle best and global best.

Step 3: Step 2 continue until termination criterion is satisfy.

Step 4: Take the global best solution as an outcome.

Figure 3.2: Steps of Swap Sequence based PSO algorithm for TSP.

3.2.5.2 Self-Tentative PSO (STPSO)

Self-Tentative PSO (STPSO) introduces tentative behavior in SSPSO that tries to improve each particle placing a node in a different position. STPSO algorithm is just like the mankind that student must have to study and understand the knowledge by himself after his learning from his teacher's. Like that particle not only needs its cognitive experience and population knowledge to adjust behavior but also need tentative behavior of its own. At the later stage of evolution, this tentative behavior is important when random adjustment operators are hard to improve the solutions [7]. The operation for tentative behavior tries to improve each particle at the end of each generation. For each particle, from the second node to the end tile following actions are done: delete the node from the original position; place it to the different positions and measure fitness values; use pointer P which records the best position. If the changes cannot improve the fitness, it can be inserted to the original position, then try to insert it to other position. If the change can improve the present fitness, then record this position in the pointer P and the fitness changes, then try other positions, and so on. After these actions, each particle would get a better position if any single node changes can improve its fitness [7]. Steps of STPSO is shown in Figure 3.3.

Step 1: Initialization: define number of particles, termination criterion, tour cost. Assign a random tour and random Swap Sequence as velocity to each of the particle. Consider previous best tour as current tour and global best tour as the best one among them.

Step 2: For each particle in the swarm

a) Calculate velocity.

b) Update solution.

c) Update particle best and global best.

Step 3: Tentative Operation on Each Particle

a) Single Node Adjustment.

b) Update particle best and global best if the new solution is superior to particle and global best respectively.

Step 4: Step 2 continue until termination criterion is satisfy.

Step 5: Take the global best solution as an outcome.

Figure 3.3: Steps of Self-Tentative PSO algorithm for TSP.

3.2.5.3 Velocity Tentative Particle Swarm Optimization (VTPSO)

Figure 3.4 shows the steps of the proposed Velocity Tentative Particle Swarm Optimization (VTPSO) to solve TSP considering the partial search technique, and a brief description of the steps of VTPSO is given below. Like other population-based algorithm. VTPSO initializes the population with random solutions and tries to improve solutions at every generation step. In initialization (Step I) VTPSO defines number of particles in the population, termination criterion, and tour cost. It also assigns a random solution (here tour) and a random velocity (here Swap Sequence) to each of the particle. At this initial stage, the previous best solution of each particle (P_i) is considered as the current random tour of it and Global best solution (G) is the best tour among them. At each iteration step, VTPSO updates the position of each particle conceiving proposed partial search-based velocity tentative behavior (Step 2 in Fig. 3.4). The velocity Swap Sequence (SS) that it calculates (Step 2.a). In Step 2.b, find the best solution. Self-

Step 1: Initialization: define number of particles, termination criterion, tour cost. Assign a random tour and random Swap Sequence as velocity to each of the particle. Consider Previous best (P_i) solution of each particle is considered as the current random tour and Global best (G) solution is the best tour among them.

Step 2: For each particle X_i in the swarm

a. Calculate tentative velocity $V_i^{(t)}$.

b. Update Solution $X_i^{(t)}$ Partial Search manner.

c. Self-Tentative Operation and Update P_i if the new solution $X_i^{(t)}$ is superior to P_i.

i) Apply Self-Tentative Operation on $X_i^{(t)}$

ii) Update P_i with new solution $X_i^{(t)}$

d. Update G if the new solution $X_i^{(t)}$ is superior to G

Step 3: Loop to **Step 2** until a termination criterion is met, usually a sufficiently good fitness or a maximum number of iterations,

Step 4: Take the global best solution G as an outcome.

Figure 3.4: Steps of Velocity Tentative PSO (VTPSO) Algorithm for TSP

Tentative (*ST*) operation of VTPSO is discussed in Step 2.c. The fitness of every new position (X_i) of a particle is checked with G and updates G if X_j is shown better than G (Step 2.d). VTPSO checks the termination criterion at the end of each iteration (in Step 3 of the Fig. 3.4) and terminates if the criterion is met. Usually, a sufficiently good fitness of G or a maximum number of iterations is considered as the termination criteria. If a termination criterion does not meet, VTPSO continues updating positions of the particles again as indicates the loop to Step 2 from Step 4 in Fig. 3.4.

3.3 Aspect of Partial Search solving TSP

Partial search seeks better result with portions of calculated tentative velocity Swap Sequence (*SS*). A *SS* is a collection of several Swap Operators (*SOs*) and considers as velocity that may apply on a solution (i.e., a tour) to get a new tour [3, 5, 8]. In the

traditional methods, all the *SOs* of a velocity *SS* applied on a tour and generate a new tour. But implementation of every successive *SO* transforms the previous tour to a new tour and reaches the final tour for the *SS*. While a traditional method ignores the intermediate tours, the partial search technique explores the option of getting better tour considering those. Therefore, the partial search enhances the capability of getting a better tour from a *SS* applying *SOs* one by one [5].

3.4 Difference between Partial Search and Local Search

Partial search is not similar to local search. Local search explores the surroundings of a solution in order to get a better candidate solution. A local search algorithm moves from solution to solution in the space of candidate solutions by applying local changes, until a solution deemed optimal is found or a time bound is elapsed. On the other hand, PS checks the intermediate solution points while the move from the old solution point towards a destined solution point.

3.5 Modified Spider Monkey Optimization Approach to Solve TSP

There are initialization process and six main steps to solve TSP using Modified Spider Monkey optimization (MSMO). The main steps are:

(1) Local Leader phase (LLP);

(2) Global Leader phase (GLP);

(3) Global Leader Learning (GLL) phase;

(4) Local Leader Learning (LLL) phase;

(5) Local Leader Decision (LLD) phase; and

(6) Global Leader Decision (GLD) phase.

Selection of Local Leaders and Global Leader, update of individual spider monkeys based on both types of leaders, updating the Local leaders and Global leader and decision phase of Local leaders and Global leader. Moreover, similar to other SI methods, initialization and termination steps are available in the proposed method. In the initialization step, a population of spider monkeys are defined with random tours; divide them into group(s), and select Local leaders and a Global leader. In every iteration, significant MSMO steps are followed and termination criteria is checked. In

the following Subsection, all the steps are discussed in details with the algorithm.

3.5.1 Initialization

In the beginning, a population of *N* spider monkeys (*SMs*) is initialized with random TSP tours and fitness of those are measured. The population may be divided into *n* groups and the local leaders (*LLs*) of individual groups are selected based on the fitness. Among the local leaders, the best one is chosen as the global leader (*GL*). MSMO starts with a single group having all the *SMs*. In such a case, *LL* and *GL* both are same *SM*. There are four control parameters which are initialized at this stage: the Maximum number of Groups (*MG*) to be allowed, perturbation rate (*pr*), Local Leader Limit (*LLL*), Global Leader Limit (*GLL*). On the other hand, different counters are also initialized in this step which are Local Leader Limit Count and Global Leader Limit Count.

3.5.2 Update of Individual SMs

In this step, each *SM* updates or changes its solution based on its current involvement, Local leader involvement and the involvement of randomly selected group members of that group. Eq. (3.5) - (3.7) are used to update a *SM* if its selected under probability of *pr*.

$$SSi = U(0,1)*(LLk - SMi) \otimes U(0,1)*(RSMr - SMi) \quad (3.5)$$

$$BSSi = Cal\_BasicSS(SSi) \quad (3.6)$$

$$SMnewi = SMi + BSSi \quad (3.7)$$

In Eq. (3.5), LL_k is the Local leader in the *kth* group, SM_i is the *ith SM*, RSM_r is another randomly selected *SM* within the group (i.e., $r \neq i$). At first, it calculates the *SS* between LL_k and SM_i; and select a portion using random *U(0,1)* for implication (say SS_1). Similarly, it calculates *SS* between RSM_r and SM_i; and select a portion for implication (say SS_2). *U(0,1)* generates a random number between *0* and *1* that is uniformly distributed. Then those two *SSs* are merged into one *SS*; i.e., $SS_i = SS_1 \otimes SS_2$. Eq. (3.6) is used to optimize SS_i through *Cal_BasicSS* function; the outcome is the basic swap sequence BSS_i. The BSS_i is applied to SM_i using Eq. (3.7) with *PS* manner explained in the previous section and get new solution $SMnew_i$. The SM_i is updated with $SMnew_i$ if

the new one is found better than the existing one. This procedure will continue for all the members of the *kth* group.

After update individual SMs considering interaction within the group, they are again interacting with global leader and another randomly selected local group member. In this case, an SM takes the decision based on the value of probability if the SM will update or not. It ensures that the SM will partake a great opportunity to make itself better than the present solution. The probability for *ith* SM is calculated according to Eq. (3.8).

$$prob(i) = 0.9 * \frac{min\_cost}{cost(i)} + 0.1 \tag{3.8}$$

Here *cost(i)* is the tour cost of *ith* SM and *min_cost* is the minimum tour cost by the best SM of that group. The equation evaluates the probability by dividing the minimum cost found so far with the cost of each SM to generate a ratio below 1. The SMs are updated in this step using Eq. (3.9) - (3.11).

$$SSi = U(0,1)*(GL - SMi) \otimes U(0,1)*(RSMr - SMi) \tag{3.9}$$

$$BSSi = Cal\_BasicSS\ (SSi) \tag{3.10}$$

$$SMnewi = SMi + BSSi \tag{3.11}$$

Here, Eq. (3.9) calculates the SS between *GL* and *SMi*, and SS of randomly selected *RSMr* and *SMi*. Eq. (3.10) is used to optimize *SSi* that yields *BSSi*; Finally, *BSSi* is applied to *SMi* using Eq. (3.11) similar to Eq. (3.7); and *SMnewi* is obtained. If the tour cost is less than the older one, then the current solution *SMi* is replaced with the new solution *SMnewi*. Otherwise, the current solution is retained.

The way of individual SMs update in two different steps considering Local leader and Global leader, respectively, are the main steps in the proposed MSMO method. However, steps have similar operations. Eq. (3.5) – (3.7) considered Local leader of the same group and a randomly selected solution while Eq. (3.9) - (3.11) considered Global leader and a randomly selected solution. The procedure of updating a *SM* based on the Local leader is demonstrated in Fig. 3.5 for better understanding. In the figure, C_1, C_2, …, C_x are the index of a tour in solutions of operating spider monkey (*SM*), Local Leader (*LL*) and Randomly selected another Spider Monkey (*RSM*). The SS generation using the experience of *SM* and *LL* is denoted by SS_{LL} = (*LL* – *SM*); similarly, SS

generation using *SM* and *RSM* is denoted by SS_{RSM} = (*RSM* – *SM*). Using *U(0,1)*, a random portion of SS_{LL} is selected, i.e., $SS_{SLL} = \{U(0,1) * SS_{LL}\}$. A portion is also selected from SS_{RSM}, i.e., $SS_{SRSM} = \{U(0,1) * SS_{RSM}\}$. SS_{SLL} and SS_{SRSM} are merged together to $SS_M = SS_{SLL} \otimes SS_{SRSM}$. Then *Cal_BasicSS()* is applied to find BSS_M. The blue circle and the green circle indicate the SOs generated by the *LL* and *RSM*, respectively. Lastly, apply SOs of BSS_M into the solution one after another and select the best solution (*SMnew*) among the tours from different for successive implications of SOs. The demonstration is shown in Fig. 3.5 is also applicable to update procedure of SM based on global leaders considering *GL* (i.e., global best solution) instead of *LL*.

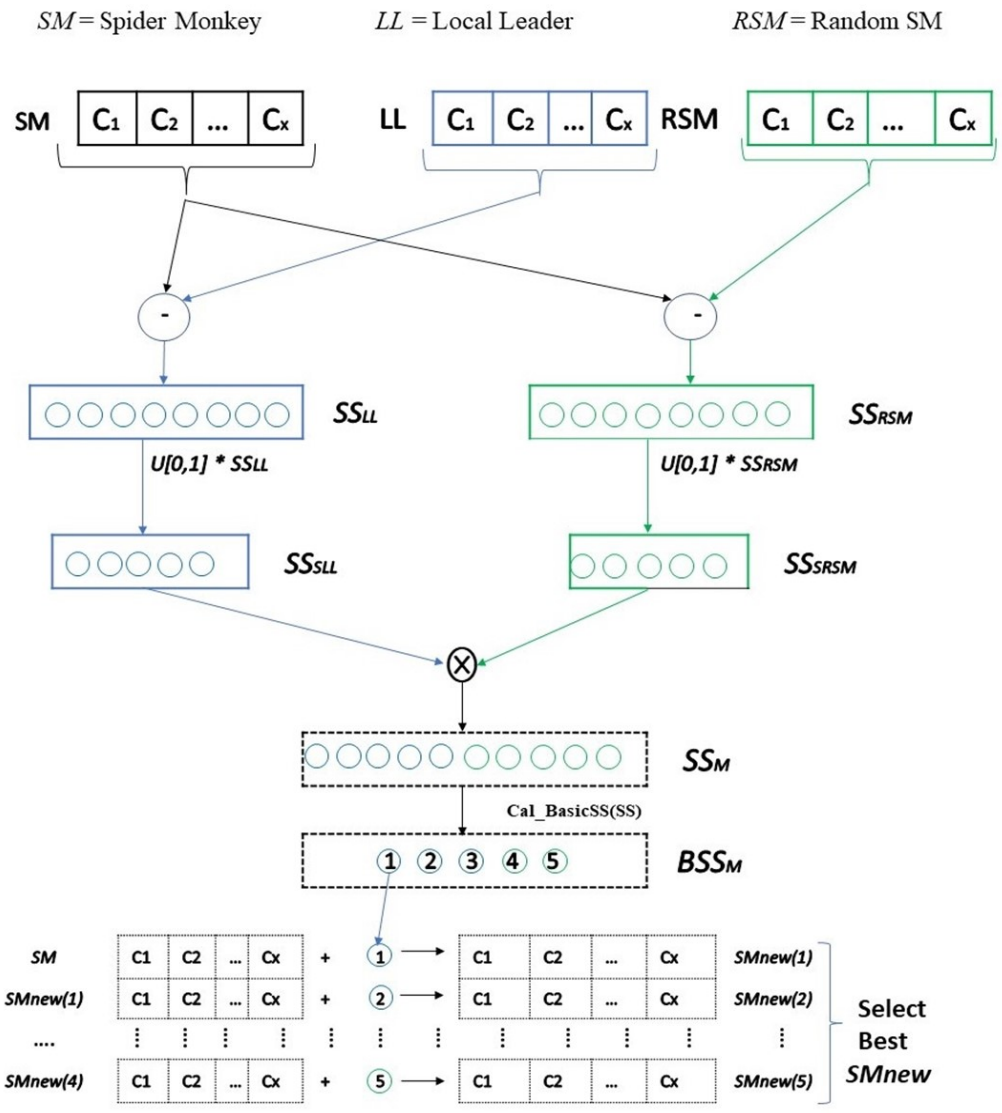

Figure 3.5: Demonstration of updating a Spider Monkey (*SM*) interacting with Local Leader (*LL*) and Random Spider Monkey (*RSM*).

3.5.3 Update of LLs and GL

In this step, the local leaders (*LLs*) and global leader (*GL*) are updated. For each group, the best SM having minimum tour cost is identified and is considered as the new Local leader if it is better than the existing group leader; otherwise, the Local leader is unchanged. If the Local leader is not modified, the value of *Local Leader Limit Count* is increased by 1.

After the update of Local leaders, the best LL (*bLL*) is identified and compared with the *GL*; update *GL* if *bLL* is better than the existing *GL*; otherwise, the Global leader is unchanged. Furthermore, *Global Leader Limit Count* is increased by 1 if the cost of *GL* is not modified.

3.5.4 Decision Phase of LL and GL

In the fourth step, if the *LL* is not updated for a certain period of time, Local leader initializes all the group's members again and the *GL* takes the decision to divide the group into subgroups or join all the groups into one group.

Local leader is supposed to update in each iteration to find a better solution. If it is not updated for *Local Leader Limit* times, all SMs of that group are updated either by randomly generated tour or interacting with the *GL* and *LL* using Eq. (3.12).

$$SMnewi = SMi + U(0,1)*(GL - SMi) + U(0,1) * (SMi - LLk) \qquad (3.12)$$

Here, *GL* is the global leader and *LLk* is the local leader of the *kth* group. Following the Eq. (3.12), *SMi* moves forward to the Global leader and detaches from the Local leader.

After checking all the local leaders limit counter, the global leader limit counter is checked. If the global leader fails to update for *Global Leader Limit* times, the Global leader forms new groups from the existing groups by dividing them until *MG* is reached. Each time *GL* divides the group, a new *LL* is selected. The global leader can choose to combine the total population into a single group if *MG* groups are formed already and no more groups are allowed.

3.5.5 Termination and Outcome

At the end of each iteration, termination or stopping criteria of the MSMO is checked. The stopping criteria can be a number of iterations, fitness threshold etc. If the termination criterion is met, the TSP solution of the Global leader is considered as the outcome of the proposed MSMO approach.

3.6 Modified Spider Monkey Optimization Algorithm

The proposed MSMO algorithm for solving TSP is shown in Table 3.2. The detailed description of the algorithm is not given since stepwise operations are described already. The notations and inputs of the proposed algorithm are listed at the beginning. Like other population-based algorithms, it initializes the TSP solutions randomly. At first, it selects the local leader *LLk* of every *kth* group and *GL* (Step 1). At each iteration step, every *SMi* will update its solution using *SS* and partial search based on *LLk* and *GL* (Step 2).

LL and *GL* will update in every iteration if the newly generated solution is better than the current solution (Step 3). *LLk* will initialize all the *SMi* if the *LLk* is not updated for a predetermined period of time and *GL* will split the population into some small sets or combine all the groups into one single set if the *GL* is not updating for a specific time (Step 4). The iteration will continue until the termination criterion (i.e., total iterations) is met (Step 5). Finally, TSP tour in *GL* is considered as the outcome of MSMO (Step 6).

Table 3.2: MSMO for Solving TSP

Input: List of Cities and their cost matrix		
I	-	Total Number of Iterations
MG	-	Allowed Maximum Group
pr	-	Perturbation Rate
LLL	-	Local Leader Limit
GLL	-	Global Leader Limit
N	-	Total Number of Spider Monkeys
Output: An optimal solution of TSP		

Variables:		
t	-	Iteration Counter
g	-	Current Number of Groups
SM	-	Population of Spider Monkey
SM_i	-	ith Spider Monkey
LL_k	-	Local Leader of kth Group
LL_{new}	-	Updated Local Leader
LL	-	List of Local Leaders
GL	-	Global Leader
SS	-	Swap Sequence of Spider Monkey
BSS	-	Basic Swap Sequence of Spider Monkey
SM_{new}	-	Updated Spider Monkey
$prob(i)$	-	Probability of ith Spider Monkey
LLL_k	-	Local Leader Limit Counter of kth Group
bLL	-	best Local Leader
GLL_c	-	Global Leader Limit Counter

Step 1: Initialization	
1.	$t \leftarrow 1$
2.	create N particles and append them to SM
3.	Assign each SM_i in SM with a random solution
4.	$g \leftarrow 1$ // Initially Consider all SM into one group
5.	Select Local Leader and Global Leader // *Both leaders are same due to single group*

Step 2: Position update of Spider Monkeys based on Local Leader and Global Leader					
1.	**for each** k^{th} *group* **do**				
2.		**for all** SM_i in k^{th} group **do**			
3.			**If** $U(0,1) >= pr$ ***then***		
4.				Calculate ***SS*** using Eq. (6)	
5.				Calculate ***BSS*** using Eq. (7)	
6.				Apply ***BSS*** into SM_i to calculate SM_{new} using Eq. (8)	
7.				**If** *fitness* (SM_{new}) > *fitness* (SM_i) *then*	
8.					$SM_i \leftarrow SM_{new}$
9.				**end if**	
10.			**end if**		
11.		**end for**			
12.	**end for**				

13.	**for each** *k^{th} group* **do**
14.	**for all** SM_i in k^{th} group **do**
15.	Calculate *prob(i)* using Eq. (9)
16.	**If** *U (0,1) <= prob(i)* ***then***
17.	Calculate ***SS*** using Eq. (10)
18.	Calculate ***BSS*** using Eq. (11)
19.	Apply ***BSS*** into SM_i to calculate SM_{new} using Eq. (12)
20.	**If** *fitness (SM_{new}) > fitness (SM_i) then*
21.	$SM_i \leftarrow SM_{new}$
22.	**end if**
23.	**end if**
24.	**end for**
25.	**end for**
Step 3: Update Phase of Local Leader and Global Leader	
1.	**for each** *k^{th} group* **do**
2.	**for all** SM_i in k^{th} group **do**
3.	calculate the fitness of SM_i
4.	**end for**
5.	$LL_{new} \leftarrow$ maximum (SM_i) *//select the highest fitness among all the spider monkeys in kth group*
6.	**If** *fitness (LL_{new}) > fitness (LL_k)* ***then***
7.	$LL_k \leftarrow LL_{new}$
8.	$LLL_k \leftarrow 0$
9.	**else**
10.	$LLL_k \leftarrow LLL_k + 1$
11.	**end if**
12.	$LL \leftarrow LL_k$
13.	**end for**
14.	$bLL \leftarrow$ maximum (LL) *//select the best local leaders heaving highest fitness among all Local Leaders*
15.	**If** *fitness (bLL) > fitness (GL)* ***then***
16.	$GL \leftarrow bLL$
17.	$GLL_c \leftarrow 0$
18.	**else**
19.	$GLL_c \leftarrow GLL_c + 1$
20.	**end if**

Step 4: Decision phase of Local Leader and Global Leader	
1.	**for each** k^{th} *group* **do**
2.	**If** $LLL_k > LLL$
3.	$LLL_k \leftarrow 0$
4.	**for all** SM_i in k^{th} group **do**
5.	**If** *U (0,1) >= pr* ***then***
6.	Initialize SM_i randomly
7.	**else**
8.	Initialize SM_i using **Eq. (13)**
9.	**end if**
10.	**end for**
11.	**end if**
12.	**end for**
13.	**If** $GLL_c > GLL$
14.	$GLL_c \leftarrow 0$
15.	**If** $g < MG$ **then**
16.	Divide the spider monkeys into $g + 1$ number of groups.
17.	**else**
18.	Disband all the groups and Form a single group.
19.	$g \leftarrow 1$
20.	**end if**
21.	**end if**
Step 5: Check the stopping criterion	
1.	**while** $t \neq I$ **do**
2.	$t \leftarrow t + 1$
3.	repeat **Step 2, 3 and 4**
4.	**end while**
Step 6: Return *GL* **as a result**	

3.7 Significance of Proposed MSMO

MSMO follows self-organization as well as separation of labor properties for finding intelligent operations for TSP. Proposed MSMO introduces SO and SS based interactions to update individual TSP tours represent by spider monkeys. In the method, a spider monkey updates its position using Local leader, Global leader, and self-experience. When the Global leader divides the groups into subgroups, it implies the divisions of labor property. Due to multiple sources consideration of SOs calculation, diverse solutions are brought in at different iterations and hence easy to find a better solution of a TSP problem.

MSMO holds a significantly different form from the existing SI methods including ACO and Velocity Tentative PSO (VTPSO). MSMO updates a SM in two different stages. In the first stage, each and every SM is updated interacting with local leader and randomly selected another SM following Eq. (3.5) – (3.7). In the second stage, each and every SM is updated interacting with Global leader and randomly selected another SM following Eq. (3.9) – (3.11). On the other hand, VTPSO as well as PSO update a particle in a single stage interaction with global best and personal best solution [7, 15]. In contrast, solutions are updated based on the pheromone on the paths in ACO that is different from other SI methods. Therefore, multiple interaction stages might be beneficial to get better outcomes with respect to other existing methods

CHAPTER 4

Experimental Studies

This chapter experimentally investigates the efficacy of proposed Modified SMO (MSMO) algorithm to solve TSP. A set of benchmark problems were chosen as a test bed and the outcome of the experiment compares to Velocity Tentative PSO (VTPSO) [15] and a prominent method Ant Colony Optimization (ACO) [68] to solve TSP. For a fair comparison, the experimental methodology is chosen carefully. Finally, an experimental analysis has been given for better understanding of the way of performance improvement in the proposed method for solving TSP.

4.1 Benchmark TSP Data and Experimental Methodology

As a test bench, a suite of 45 benchmark problems are considered from TSPLIB [16], the well-known benchmark repository for TSPs. The size of selected problems varies from 14 to 493 which gives a diverse test bed. A numeric value with a problem name indicates the number of cities in the problem. Like st70 and rat99 have 70 and 99 cities, respectively. In the dataset, coordinates of individual cities are given for a particular problem which needs to process to use any system. The cost matrix is prepared using coordinates of cities which is then used to calculate the tour cost. A tour cost of a TSP solution is the accumulation of the cost distances of individual links considered in the tour. Tour cost is considered as the fitness value in solving TSP by any optimization method and tour cost minimum is good.

The parameters of MSMO are maximum group (*MG*); *Local Leader Limit* (*LLL*) and *Global Leader Limit* (*GLL*); and perturbation rate (*pr*). In the experiment, *MG* is set as 5; both *LLL* and *GLL* is set to 100; *pr* is initialized with 0.1. In ACO, alpha and beta were set to 1 and 3 respectively. The selected parameters were not optimal values but considered for simplicity as well as for fairness in comparison.

The methods ACO, VTPSO and proposed MSMO are implemented on Visual C++ of Visual Studio 2017 on Windows 10 OS environment. Tests have been led on a desktop computer (HP ProBook 450 G1, Intel(R) Core (TM) i5-4200M CPU @ 2.50 GHz, RAM 12.0 GB).

4.2 Experimental Analyses varying Population Size and Total Iteration

Proposed MSMO is a novel population-based SI method; therefore, like other SI methods, population size *M* (here number of monkeys in population) and total iteration are the two most important parameters to get appropriate outcome. In this section, the effects of population size and total iteration on MSMO are investigated on three selected problems which are berlin52, rat99 and gr137. The selected problems are different in size and are studied many existing studies. ACO and VTPSO are also included in the analyses for a better understanding of MSMO compared to the algorithms.

4.2.1 Impact of Population Size Variation

This section compares the efficiency of ACO, VTPSO and proposed MSMO for varying population size. The population size considered for ACO/VTPSO from 10 to 500. In contrary, the population size started in MSMO from 20 to manage MG as 5. For a fair comparison, fixed 500 iterations were considered for all three methods.

Figure 4.1 presents the tour cost for different population size on berlin52, rat99 and gr137 problems. The presented results are the average of 10 individual runs. Standard Deviation (SD) values of individual results are shown as vertical line bars on the average value. It is noticeable from the figure that VTPSO and MSMO performed very badly at small population size (e.g., 10, 20) and improved with increasing the population size. Meanwhile, ACO is shown an almost unchanged performance for a specific problem. In ACO, each ant starts from a city and initially pheromone are equal in all the path; therefore, ants larger than total city contains several duplicate paths and failed to improve solution at all. On the other hand, due to pheromone-based heuristic, ACO is shown more contestant results over different runs; therefore, SD values of ACO are comparatively very low with respect to the values of VTPSO and MSMO.

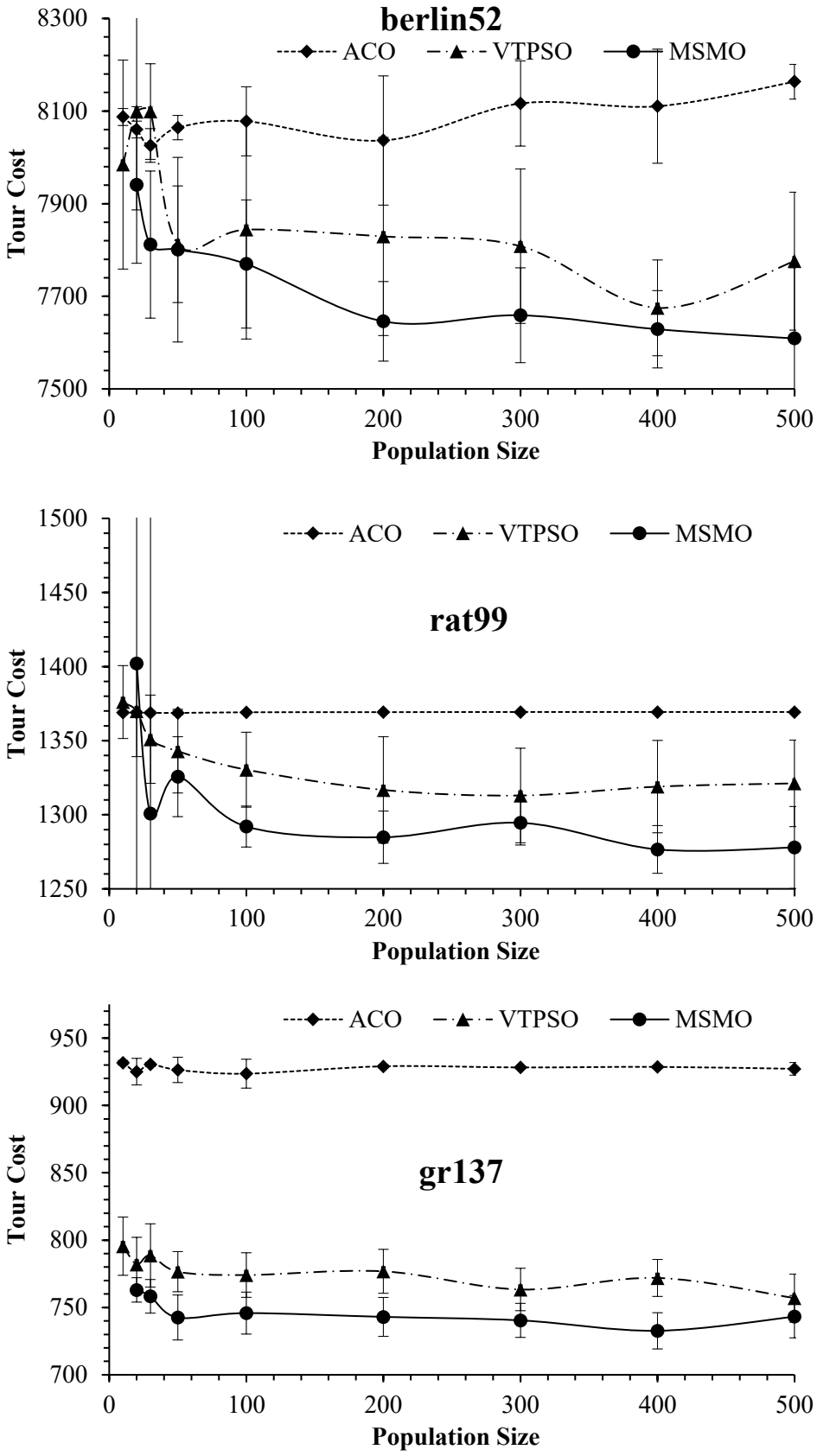

Figure 4.1: Impact of population size on tour cost.

It looks interesting from Figure 4.1 that MSMO is the best in general for any population size for all three problems. As an example, for the gr137 problem ACO, VTPSO, and MSMO showed tout costs of 925.05, 782.03, and 763.03, respectively when the population size was 20. The tour costs became 923.57, 774.07, and 737.44 for population size increased to 100. On the other hand, VTPSO and MSMO achieved best tour costs of 756.96 and 724.46, respectively at population size 500 and 300, respectively. Comparing results of all the problems, MSMO seems to perform well with population variation outperforming ACO/VTPSO. At a glance, proposed MSMO has shown better results in different population sizes.

4.2.2 Impact of Total Iteration Variation

This section represents the comparative performance analysis among ACO, VTPSO, and MSMO with respect to the total iteration number. For a fair comparison, the iteration number was varied from 10 to 1000 maintaining the fixed population at 100 for all three methods.

Figure 4.2 demonstrates the tour costs of ACO, VTPSO, and MSMO for different iterations on berlin52, rat99 and gr137 problems. The presented results are the average of 10 individual runs with SD values. It is observed from the figure that ACO was most invariant to iteration variation. On the other hand, tour costs were shown very high for a small number of iterations (e.g., 20, 50) and improved with the increase of iteration number for both MSMO and VTPSO. It is also notable that after certain iteration value, the changing result was not significant for VTPSO and MSMO.

It is notable from Fig. 4.2 that although both MSMO and VTPSO were found competitive for a small number of iterations, MSMO outperformed VTPSO and ACO in general. As an example, for the berlin52 problem at iteration 10, ACO, VTPSO, and MSMO achieved tour costs of 8102.24, 8407.58, and 8353.17, respectively. For the same berlin52 problem, the best tour costs for ACO and VTPSO were 8064.98 (at 200 iteration) and 7770.38 (at iteration 600), respectively. On the other hand, MSMO was shown 7656.44 at iteration 800 and outperformed ACO/VTPSO and the achieved tour cost is much better than the best-achieved value of ACO. The similar performance of MSMO for rat99 and gr137 problems reveal that proposed MSMO is an effective method for solving TSP. In short, MSMO has demonstrated the capacity to accomplishing better outcome while varying iteration number.

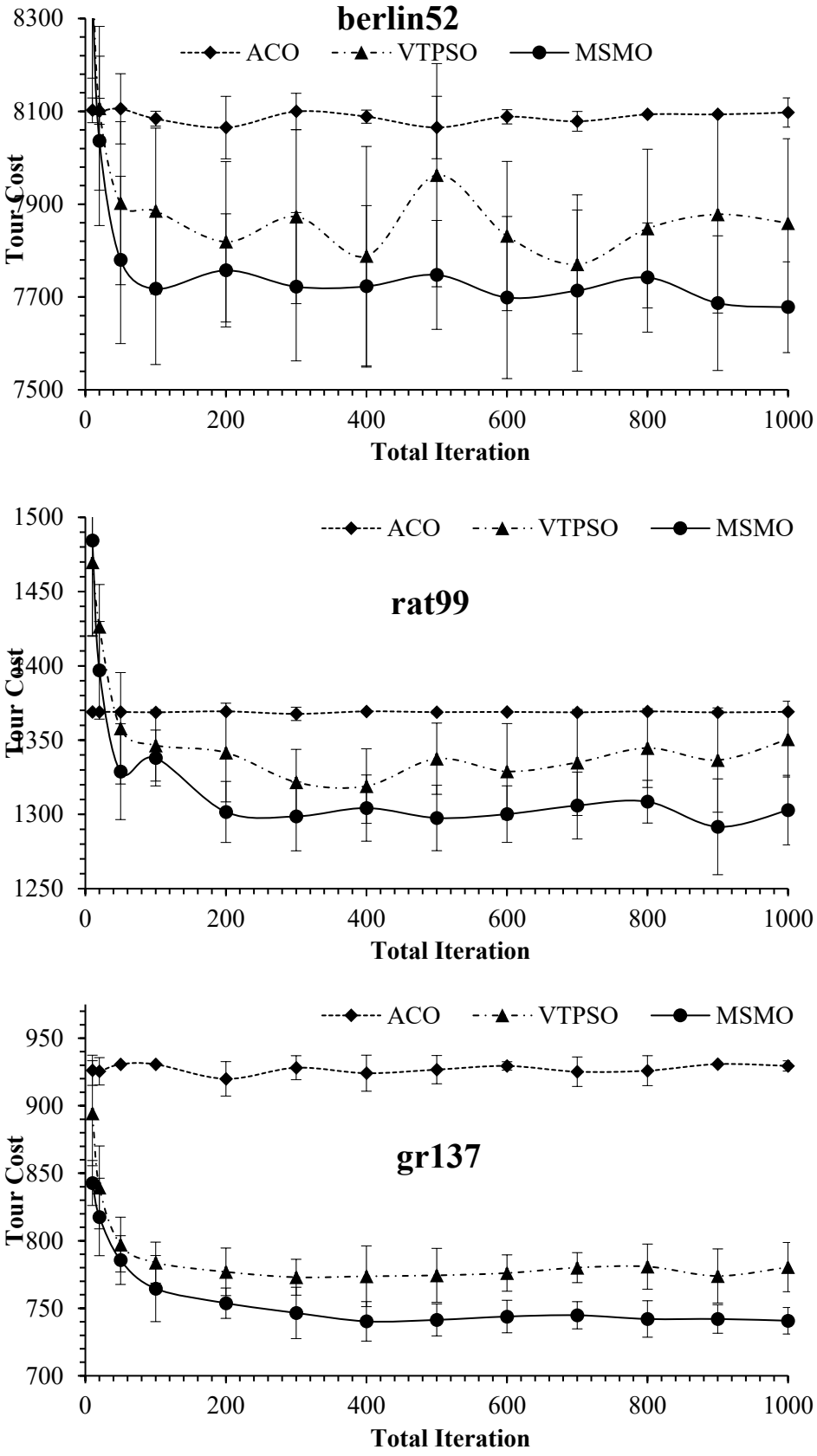

Figure 4.2: Impact of total iteration on tour cost.

4.3 Performance Comparison with ACO and VTPSO

The experimental results of the proposed MSMO in solving the suite of 45 benchmark TSPs are compared with ACO and VTPSO. For the fairness in comparison, the population size was considered in between 100 and 300 since outcomes did not change significantly better for larger population size. On the other hand, the iteration was fixed at 500 as termination criteria based on the observation from iteration variation of the previous section. The best outcome for different population sizes (i.e., 100, 200 and 300) for particular a problem by a method is considered in comparison.

Table 4.1 and Table 4.2 compares the performance of ACO, VTPSO, and MSMO on 20 different runs for each TSP instance. Table 4.1 compare the performance of small sized (up to 150) benchmark TSPs and Table 4.2 compares the performance of large sized benchmark TSPs. For a better evaluation, SD value of 20 runs is placed with average value for a particular problem. Since the best solution heaving minimum tour cost from different runs may use as the outcome, minimum tour cost from different runs are also included in performance evaluation and placed in the Table 4.1 and Table 4.2. For both average and minimum tour costs, the best value (i.e., smallest cost) among the three methods was shown in bold-face type and the worst value (i.e., biggest cost) was indicated by underlined face type. The average on all the instances and best/worst count were shown at the bottom of the two table. It indicates how many problems instances a method gave the best/worst result. A pairwise summary Victory-Draw-Defeat from Table 4.1 and Table 4.2 for all the methods are also presented in Table 4.3 for pairwise algorithm comparison.

On the basis of the average tour cost in Table 4.1 and Table 4.2 for all the problems, MSMO is the best and ACO is the worst among the three methods. The average tour cost of MSMO was 17233.6 in Table 4.1 and 33228.1 in Table 4.2. The accomplished average tour costs of ACO and VTPSO were 18729.5 and 17393.1 in Table 4.1 and 35787.7 and 33794.7 in Table 4.2 respectively. In Best/Worst count for individual problems, MSMO showed best results in 21 cases out of 25 problems in Table 4.1 and 17 cases out of 20 problems in Table 4.2. VTPSO showed the best result for 8 cases in Table 4.1 and 3 cases in Table 4.2. On the other hand, in Table 4.1 ACO showed the best result for only gr17 which is 2332.58. In Table 4.1, the problem gr17 is small in size heaving 17 cities only and both VTPSO and MSMO showed same tour cost of

2332.6 which is very competitive to ACO. In general, both VTPSO and MSMO showed competitive performance for small sized problems (e.g., gr24, fri26) but MSMO showed significantly better performance for large sized problems. In Table 4.2, MSMO showed best tour costs for all the problems heaving problem size larger than 200 (i.e., gr202 to d493). In pair Victory-Draw-Defeat comparison in Table 4.3, MSMO outperformed ACO and VTPSO in 44 and 33 cases, respectively. MSMO was inferior to VTPSO for only six cases and the results of remaining six cases were same for both MSMO and VTPSO.

For solving TSP, ACO is a well-studied prominent SI method. Since it starts placing different ants in different cities, its initialization did not differ much among individual runs. Therefore, the tour costs of ACO in different runs were found consistent showing lower SD values with respect to other prominent existing methods including VTPSO [15]. In Table 4.1, for small problems (where the number of cities is few) like ulysses16, bays29 the tour cost for all 20 individual runs are the same, so the SD of the average tour cost was zero. On the other hand, VTPSO gives most variant outcomes among different runs showing the largest SD value. But it is interesting to observe from the results presented in Table 4.1 and 4.2 that, the SD value of MSMO is lower than VTPSO in general. As an example, in Table 4.1 ACO achieved an average tour cost of 1369.2 with SD of 0.85 in case of rat99. For the same problem, VTPSO and MSMO achieved average tour costs 1325.8 (with SD 34.83) and 1291.93 (with SD 21.07), respectively.

The best result from the different runs may use the outcome of a method. Therefore, the minimum tour cost achieved from 20 runs were also compared in Table 4.1 and 4.2 for better understanding. On the basis of the minimum tour costs, MSMO showed a better result than ACO and VTPSO in both Table 4.1 and 4.2. MSMO demonstrated the average minimum tour cost of the 25 problems (i.e., 16638.4) in Table 4.1 and 20 problems (i.e. 32230.3) in Table 4.2 and both the value is better than ACO and VTPSO. The average lowest tour costs of VTPSO and ACO were 35440.2 and 18651.9 in Table 4.1 and 32240.7 and 16816.4 in Table 4.2, respectively. Based on the Best/Worst summary, MSMO achieved the best result for 24 cases in Table 4.1 and 13 cases in Table 4.2, but none of the cases shown the worst result. On the other hand, VTPSO shows 11 best results in Table 4.1 and 7 best results in Table 4.2. Furthermore, according to the Victory-Draw-Defeat summary, ACO was worse than MSMO in all the cases except gr17. For gr17, all the methods achieved same tour cost of 2332.58 as

Table 4.1: Performance comparison among ACO, VTPSO and MSMO to solve Small Sized (up to 150) Benchmark TSPs.

SL	Problems	Average Tour Cost (Standard Deviation)			Minimum Tour Cost		
		ACO	VTPSO	MSMO	ACO	VTPSO	MSMO
1	burma14	31.21 (0)	**30.87** (0)	**30.87** (0)	31.21	**30.87**	**30.87**
2	ulysses16	77.13 (0)	**73.99** (0)	**73.99** (0)	77.13	**73.99**	**73.99**
3	gr17	**2332.58** (0)	2332.6 (0)	2332.6 (0)	**2332.58**	**2332.58**	**2332.58**
4	gr21	2953.46 (2.67)	**2672.3** (0)	**2672.3** (0)	2949.81	**2672.27**	**2672.27**
5	ulysses22	86.71 (0.45)	**75.33** (0.06)	75.4 (0.02)	84.78	**75.31**	**75.31**
6	gr24	1267.13 (0)	**1249.8** (0)	**1249.8** (0)	1267.13	**1249.82**	**1249.82**
7	fri26	646.48 (0)	**635.58** (0)	**635.58** (0)	646.48	**635.58**	**635.58**
8	bayg29	9964.78 (0)	9078 (17.61)	**9074.2** (0)	9964.78	**9074.15**	**9074.15**
9	hk48	12733 (78.99)	11376 (281.4)	**11296** (215.76)	12699.86	**11104.67**	**11104.67**
10	eil51	504.97 (1.86)	439.87 (5.65)	**436.96** (4.73)	499.92	429.51	**428.86**
11	berlin52	8078.82 (49.05)	7728 (161.8)	**7633.6** (85.4)	7870.45	**7544.37**	**7544.37**
12	st70	749.19 (8.28)	711.5 (14.57)	**702.64** (15.04)	734.19	682.57	**677.11**
13	eil76	598.99 (8.22)	**569.1** (9.12)	572.7 (7.56)	581.42	559.25	**558.68**
14	pr76	127181.5 (172.01)	112549.2 (1420)	**111299.3** (2050.48)	127025.9	109586.1	**108159.4**
15	gr96	588.92 (7.93)	537.92 (11.89)	**530.45** (7.29)	563.77	**515.27**	518.38
16	rat99	1369.2 (0.85)	1325.8 (34.83)	**1291.93** (21.07)	1366.3	1256.25	**1225.56**
17	kroa100	24659.09 (66.87)	22400.48 (529.13)	**22024.27** (508.89)	24504.9	21307.44	**21298.21**
18	kroB100	24937 (284.08)	23236.43 (522.63)	**23022.37** (277.32)	24664.13	22475.67	**22308**
19	rd100	9406.56 (91.14)	8502.73 (167.7)	**8377.76** (209.4)	9120.92	8094.75	**8041.3**
20	eil101	732.86 (7.48)	679.14 (13.29)	**674.4** (10.97)	715.35	653.16	**648.66**
21	lin105	15785.33 (421.1	15684.64 (610.38)	**15114** (500.76)	15364.58	14581.58	**14383**
22	pr107	46535 (129.95)	**45287.9** (1207.52)	45666.99 (1300.43)	46317.71	44436.25	**44385.86**
23	pr124	65145 (0)	63939.97 (1739.4)	**62443.49** (1644.93)	65145.25	61076.73	**60285.21**
24	pr136	110946.3 (322.83)	102945.7 (1688.2)	**102872** (2855.28)	110872.2	99247.01	**97538.68**
25	gr137	927.44 (9.99)	765.21 (22.25)	**736.67** (15.61)	896.07	714.18	**709.48**
Average		**18729.5**	**17393.1**	**17233.6**	**18651.9**	**16816.4**	**16638.4**
Best/Worst		**1/24**	**8/1**	**21/1**	**1/24**	**11/0**	**24/0**

Table 4.2: Performance comparison among ACO, VTPSO and MSMO to solve Large Sized Benchmark TSPs.

SL	Problems	Average Tour Cost (Standard Deviation)			Minimum Tour Cost		
		ACO	VTPSO	MSMO	ACO	VTPSO	MSMO
1	kroA150	31170.42 (365.45)	**28262.98** (455.95)	28354.09 (524.91)	30546.06	**27232.1**	27591.44
2	kroB150	29922.94 (961.65)	27987.85 (620.72)	**27576.16** (625.26)	29124.59	**26579.73**	26601.94
3	pr152	79423 (198.43)	**76272.37** (1199.9)	76526.77 (1663.08)	79153.02	74414.17	**74243.91**
4	u159	47615.78 (242.53)	45441.55 (1178.06)	**42598.3** (0)	47514.43	43579.82	**42598.3**
5	rat195	2555.3 (26.41)	2554.1 (47.56)	**2488.55** (50.48)	2534.83	2452.92	**2372.89**
6	d198	17374 (111.15)	16489.28 (206.79)	**16270.47** (171.2)	17301.47	16066.44	**15978.13**
7	kroA200	34548 (0)	**31502.33** (531.41)	31828.64 (652.32)	34547.69	30602.81	**30481.35**
8	kroB200	35109.8 (280.51)	31923.15 (553.02)	**31781.62** (487.39)	34207.79	30767.52	**30716.5**
9	gr202	554.58 (6.19)	513.04 (7.24)	**508.81** (4.08)	545.33	**497.02**	501.83
10	tsp225	4544.82 (54.17)	4210.65 (66.44)	**4162.79** (66.08)	4396.39	4095.01	**4013.68**
11	pr226	90501.47 (0.02)	88031.31 (3461.96)	**85935.69** (2105.13)	90501.46	**81050.23**	83587.98
12	gr229	1915.45 (21.82)	1736.35 (26.19)	**1730.46** (20.05)	1865.63	**1676.46**	1683.45
13	gil262	2787.01 (31.83)	2631.69 (39.44)	**2627.87** (42.39)	2730.52	2547.16	**2543.15**
14	pr299	57509.4 (444.01)	53154.34 (1389.36)	**51747.99** (863.32)	56700.65	**50571.83**	50579.82
15	lin318	48583.74 (560.27)	46209.53 (773.01)	**45460.25** (660.47)	47442.95	44724.38	**44118.66**
16	linhp318	48333.33 (415.2)	46329.87 (948.05)	**45730.57** (929.73)	47577.77	44337.02	**43831.44**
17	fl417	13618.22 (209.51)	12980.24 (311.88)	**12950.77** (360.99)	13296.85	12376.53	**12218.98**
18	gr431	2374.72 (16.03)	2061.72 (30.34)	**2042.77** (24.08)	2334.63	2021.95	**1993.15**
19	pr439	127523 (224.55)	119442.2 (2770.58)	**116379.2** (2462.82)	127228	**112088**	112105.2
20	d493	39789 (407.83)	38159.18 (603.84)	**37861.14** (426.97)	39254	37132.09	**36844.63**
Average		**35787.7**	**33794.7**	**33228.1**	**35440.2**	**32240.7**	**32230.3**
Best/Worst		**0/20**	**3/0**	**17/0**	**0/20**	**7/0**	**13/0**

seen in Table 4.1. In Table 4.3, between MSMO and VTPSO, MSMO was better than VTPSO for 27 cases but MSMO was inferior to VTPSO eight cases; and rest 10 cases both the methods give the same result. Finally, MSMO might be a good choice to solve TSP considering the results presented in Table 4.1 and 4.2 and summarized in Table 4.3.

Table 4.3. Pairwise Victory-Draw-Defeat Summary of Results Presented in Table 4.1 and 4.2

(A) Summary on Average Tour Cost

Method	**ACO**	**VTPSO**	**MSMO**
ACO	-	44-0-1	44-0-1
VTPSO		-	33-6-6

(B) Summary on Minimum Tour Cost

Method	**ACO**	**VTPSO**	**MSMO**
ACO	-	44-1-0	44-1-0
			27-10-8

CHAPTER 5

Conclusion

Optimization has been an active area of research for several decades. Traveling Salesman Problem (TSP) is one of the popular combinatorial problem and many new techniques are proposed recently to solve it. Recently nature inspired population based methods had drawn great attraction to solve TSP. This thesis investigates a new Swap Operator (SO) based Spider Monkey Optimization (SMO) method for solving TSP. This chapter will now give a short summary of the main points described in this thesis. Also, it discusses possible future works based on the outcome of the present work

5.1 Achievement

Standard SMO is a recently developed SI method for function optimization. Modified Spider Monkey Optimization (MSMO) is developed modifying basics of SMO as well as introducing new operations to solve TSP. In MSMO, individual spider monkey represents a TSP solution. In every iteration, the spider monkey tries to update considering Local leader and Global leader solutions. Swap Sequence (SS) and Swap Operator (SO) related procedure is developed and applied to improve TSP solutions of individual monkeys. Grouping spider monkeys and updating a monkey interacting with group leader (i.e., best within a group) and Global leader (i.e., best among all the groups) in two different stages is the significance property of MSMO. At first, SS with several SOs is measured for a particular monkey and the best solution is considered for implication of several or all SOs of the SS. MSMO has been tested on a large suite of TSPs from benchmark repository and revealed as a best suited method for solving TSP. The proposed MSMO is shown to produce optimal solution within a minimal time than conventional methods such VTPSO in solving benchmark TSPs. The reason behind the less time requirement is revealed from the experimental analysis is that MSMO converge faster due to intermediate tour evaluation.

5.2 Future study

Idea of solving TSP in MSMO based on the intelligent behavior of spider monkeys is opened a new direction to solve other discrete optimization tasks. Solving different scheduling and routing problems extending conceiving ideas from MSMO might be interesting and remained as a future study. This study considered partial search maintaining sequence of SOs in the SS and identifies that a portion of SS may give better than whole SS implementation. It is notable that SOs may be applied independently without sequence. Such consideration may give better result.

REFERENCES

[1] J.C. Bansal, H. Sharma, S.S. Jadon, and M. Clerc, “Spider Monkey Optimization for numerical optimization,” Memetic Computing, vol. 47, pp. 31-48, 2014.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz, “Swarm intelligence: from natural to artificial systems,” Oxford University Press, New York, 1999.

[3] M. A. H. Akhand, S. Akter, S. S. Rahnian, and M. M. H. Rahman, "Particle Swarm Optimization with Partial Search to Solve Traveling Salesman Problem," International Conference on Computer and Communication Engineering, Kuala Lumpur. Malaysia, 3-5 July 2012.

[4] M.M. Symington, “Fission-fusion social organization in Ateles and pan,” International Journal Primatology, vol. 11 pp. 47–61, 1990.

[5] K. P. Wang, L. Huang, C. G. Zhou, and W. Pang, “Particle swarm optimization for traveling salesman problem,” in Proc. International Conference on Machine Learning and Cybernetics, pp. 1583–1585, November 2003.

[6] X. Wei, Z. Jiang-Wei, and Z. Hon-lin, “Enhanced Self-Tentative Particle Swarm Optimization Algorithm for TSP,” Journal of north China electric power university, vol. 36, no. 6, pp. 69–74, 2009.

[7] J. Zhang and W. Si, “Improved Enhanced Self-Tentative PSO Algorithm for TSP,” in Proc. Sixth IEEE International Conference on Natural Computation 2010, Yantai, Shandong, August 2010, pp. 2638–2641.

[8] S. Cong, Y. Jia, and K. Deng, "Particle Swarm and Ant Colony Algorithms and Their Applications in Chinese Traveling Salesman Problem," New Achievements in Evolutionary Computation, Book edited by: Peter Korosec, In-Tech Publisher, ISBN 978-953-307-053-7, pp. 318, February. 2010.

[9] X. Geng, Z. Chenb, W. Yanga, D. Shia, and K. Zhaoa, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search," Journal of Applied Soft Computing, Elsevier Publisher, pp. 3680-3689, February 2011.

[10] R. Gan, Q. Guo, H. Chang, and Y. Yi. "Improved ant colony optimization algorithm for the traveling salesman problems." Journal of Systems Engineering and Electronics vol. 21.No. 2, pp.329-333. April 2010.

[11] R. Matai, S.P. Singh, and M.L. Mittal, “Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches,” Traveling Salesman Problem, Theory and Applications, Edited by D. Davendra, Intech, pp 1-24, 2010.

[12] Y. F. Liao, D. H. Yau, and C. L. Chen, "Evolutionary algorithm to traveling salesman problems," International Journal of Computers & Mathematics with Applications, Elsevier Publisher, December 2011.

[13] P. Merz and T. Fischer, "A Memetic Algorithm for Large Traveling Salesman Problem Instances." The Seventh Metaheuristics International Conference. Montreal. Canada, June 2007.

[14] S. M. Chen and C.Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques." Journal of Expert Systems with Applications. Elsevier Publisher, pp. 14439-14450, 2011.

[15] M.A.H. Akhand, S. Akter, M.A. Rashid, and S.B. Yaakob, “Velocity Tentative PSO: An Optimal Velocity Implementation based Particle Swarm Optimization to solve Traveling Salesman Problem,” International Journal of Computer Science, vol. 42, July 2015.

[16] TSPLIB – a library of sample symmetric Traveling Salesman Problem instance. 1995 Available: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/ index.html/.

[17] O. Cordon, F. Herrera, and T. Stutzle, “A review on the ant colony optimization metaheuristics: basic, models and new trends,” Math ware and Soft Computing, vol. 9, pp. 141-175, 2002.

[18] M.A.H. Akhand, P.C. Shill, Md. Forhad Hossen, A.B.M. Junaed, and K. Murase, “Producer-Scrounger Method to solve Traveling Salesman Problem,” I.J. Intelligent Systems and Applications, vol. 7, pp. 29-36, 2015.

[19] K.L. Hoffman, M. Padberg, and G. Rinaldi, “Traveling Salesman Problem,” Encyclopedia of Operations Research and Management Science, pp. 1573-1578, 2013.

[20] J.K. Lenstra and H.G. Rinnooy Kan, “Some simple applications of the Traveling Salesman Problem,” Operational Research Quarterly, vol. 26, pp. 717-733, 1975.

[21] Sandeep Kumar, Rajani Kumar, and Vivek Kumar Sharma, “Fitness Based Position update in Spider Monkey Optimization algorithm,” International Conference on Soft Computing and Software Engineering, pp. 442-449, 2015.

[22] K. Lenin, B. Ravidhranath, and M. Suryakalavathi, “Modified Monkey Optimiztion algorithm for solving optimal reactive Power Dispatch Problem,” Indonesian Journal of Electrical Engineering and Informatics, vol. 3, pp. 55-62, 2015.

[23] U.Singh, R. Salgotra, and m. Rattan, “A Novel Binary Spider Optimization Algorithm for Thinning of Concentric Circular Antenna Arrays,” IETE journal of Research, 2016.

[24] Tina Gui, Christopher Ma, F. Wang, J. Li, and Dawn E. Wilkins, “A Novel Cluster-based Routing Protocol Wireless Sensor Networks using Spider Monkey Optimization,” 42nd Annual Conference of the IEEE Industrial Electronics Society, pp. 5657-5662, 2016.

[25] Avinash Sharma, Akshay Sharma, BK Panigrahi, Deep Kiran, and Rajesh Kumar, “Ageist Spider Monkey Optimization Algorithm,” Swarm and Evolutionary computation, 2016.

[26] N. Nayak, M.S. Mahali, I. Majumder, and R.K. jena, “Dynamic Stability Improvement of VSC-HVDC Connected Multi machine Power System by Spider Monkey optimization Based PI controller,” International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), pp. 152-157, 2016.

[27] G. Hazrati, H. Sharma, N. Sharma, and J.C. Bansal, “Modified Spider Monkey Optimization,” International Workshop on Computational Intelligence, pp. 209-214, 2016.

[28] U. Singh and R. Salgotra, “Optimal Synthesis of Linear Antenna Arrays Using Modified Spider Monkey Optimization,” Arabian Journal for Science and Engineering, vol. 31, pp. 2957–2973, 2016.

[29] Ali A. Al-Azza, Ammar A. Al-jodah, and Frances J. Harackiewicz, “Spider Monkey Optimization (SMO): A Novel Optimization Technique in Electromagnetics,” IEEE Radio and Wireless Symposium (RWS), pp. 238-240, 2016.

[30] R. Rohila, V. Sikri, and R. Kapoor, “Spider Monkey Optimization assisted particle filter for robust object tracking,” IET journal, pp. 207-219, 2016.

[31] Ali A. Al-Azza, Ammar A. Al-jodah, and Frances J. Harackiewicz, “Spider Monkey Optimization: A Novel Technique for Antenna Optimization,” IEEE Antennas and Wireless Propagation Letters, 2016.

[32] K. Gupta, Kusum Deep, and J.C. Bansal, “Spider monkey optimization algorithm for constrained optimization problems,” Soft Computing, vol. 21, pp. 6933–6962, 2017.

[33] N. Mittal, U. Singh, R. Salgotra, and B.S. Sohi, “A Boolean Spider Monkey Optimization based energy efficient clustering approach for WSNs,” Wireless Network, vol. 24, pp. 2093–2109, 2018.

[34] V. Kaur, K. Singh, and G. Kaur, “Analysis of Adaptive Lossless Image Compression using Hybrid Particle Swarm Optimization Spider Monkey Optimization (PSOSMO) Algorithm,” International Journal of Engineering Science and Computing, pp. 10979-10983, 2017.

[35] B. Omkar, D. Preet, D. Swarada, and D. Poonam, “Dengue Fever classification using SMO Optimization Algorithm,” International Research Journal of Engineering and Technology (IRJET), vol. 4, pp. 1683-1686,2017.

[36] Joydip Dhar and Surbhi Arora, “Designing fuzzy rule base using Spider Monkey Optimization Algorithm in cooperative framework,” Future Computing and Informatics Journal, pp. 1-8, 2017.

[37] V. Hiranandani, S. Sharma, and Ajay Saini, “Spider Monkey Optimization Algorithm: Estimation of Frequency-Modulated (FM) Sound Waves,” International Journal of Electrical Electronics & Computer Science Engineering, pp. 111-116, 2018.

[38] D. Tripathy, B. K. Sahu, B. Patnaik, and N.B.D. Choudhury, “Spider Monkey Optimization Based Fuzzy-2D-PID Controller for Load Frequency Control in Two-Area Multi-Source Interconnected Power System,” International Conference on Technologies for Smart-City Energy Security and Power (ICSESP), March 2018.

[39] R. C. Eberhart, and Y. Shi, "Particle swarm optimization: developments, applications, and resources," In Proc. of the Conference on Evolutionary Computation, Seoul. South Korea, pp. 81 - 86, 27-30 May 2001.

[40] X. F. Xie. W. J. Zhang, and Z. L. Yang. "Adaptive Particle Swarm Optimization on Individual Level," International Conference on Signal Processing, China, 2002.

[41] X. Yan, C. Zhang, W. Luo, W. Li. W. Chen, and H. Liu. "Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm," International Journal of Computer Science Issues, vol. 9, issue 6, no. 2, pp. 1694-1714, November 2012.

[42] H. Afaq and S. Saini, "On the Solutions to the Travelling Salesman Problem using Nature Inspired Computing Techniques," International Journal of Computer Science Issues, vol. 8, issue 4, No 2, July 2011.

[43] M. Yuanbin, X. Shuihua, and L. Jizhong, "Particle Swarm Optimization with Complex Local Searching for Solving Optimal Moving Path for PCB," Advances in Mathematical and Computational Methods, vol. 1, no. 1, September 2011.

[44] A. J. Ouyang, Y. Q. Zhou, "An Improved PSO-ACO Algorithm for Solving Large- Scale TSP." Advanced Materials Research. vol. 143, pp. 1154-1158. January 2011.

[45] R. F. Abdel-Kader, "Fuzzy Particle Swarm Optimization with Simulated Annealing and Neighborhood Information Communication for Solving TSP." International Journal of Advanced Computer Science and Applications, vol. 2. no. 5, 2011.

[46] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large scale traveling salesman problem," Operations Research, 2:393–410, 1954.

[47] P. Miliotis, "Using cutting planes to solve the symmetric Travelling Salesman problem," Mathematical Programming, vol. 15, pp. 177-188, 1978.

[48] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "An analysis of several heuristics for the traveling salesman problem," SIAM Journal of Computing, vol. 6, pp. 563-581, 1977.

[49] G. Reinelt, "The traveling salesman: Computational solutions for TSP applications," Springer Verlag, Berlin, Germany, 1994. LNCS 840.

[50] D.S. Johnson and L.A. McGeoch, "The TravelingSalesman Problem: A Case Study in Local Optimization," November 20, 1995.

[51] http://en.wikipedia.org/wiki/Hill_climbing

[52] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," Journal of Chemical Physics, vol. 21, no. 6, pp. 1087–1092, 1953.

[53] F. Glover, “Heuristics for integer programming using surrogate constraints,” Decision Sciences, vol. 8, no. 1, pp. 156–166, 1977.

[54] M. Dorigo, V. Maniezzo, and A. Colorni, “The ant system: Ant autocatalytic optimizing process,” Technical Report TR91-016, Politenico di Milano, 1991.

[55] J. H. Holland, “Adaptation in Natural and Artificial Systems,” MIT Press, 1975.

[56] C. R. Carpenter, “Behavior of Red Spider Monkeys in Panama,” Journal of Mammalogy, vol. 6, no. 3, pp. 171–180, 1935.

[57] R. O. Deaner, C. P. Chaik, and V. E. Johnson, “Do some taxa have better domain-general cognition than others? A meta-analysis,” Evolutionary Psychology, vol. 4, pp. 149–196, 2006.

[58] V. Agrawal, R. Rastogi, and D. C. Tiwari, “Spider Monkey Optimization: a Survey,” International Journal of System Assurance Engineering and Management, vol. 9, pp. 929-941, August 2018.

[59] A. Sharma, H. Sharma, A. Bhargava, and N. Sharma, “Power law-based local search in spider monkey optimization for lower order system modeling,” International Journal of Systems Science, vol. 48, no. 1, pp. 150–160, 2016.

[60] R. Cheruku, D. R. Edla, V. Kuppili, “SM-RuleMiner: Spider monkey-based rule miner using novel fitness function for diabetes classification,” Computers in Biology and Medicine, vol. 81, pp. 79–92, 2017.

[61] A. Kaur, “Comparison analysis of CDMA multiuser detection using PSO and SMO,” International Journal of Computation Application, vol. 133, no. 2, pp. 47–50, 2016.

[62] K. Selvam, D. V. Kumar, “Frequency control of microgrid with wind perturbations using levy walks with spider monkey optimization algorithm,” International Journal of Renewable Energy Research, vol. 7, pp. 146–156, 2017.

[63] A. F. Ali, “An improved spider monkey optimization for solving a convex economic dispatch problem,” Nature-inspired computing and optimization, Springer, pp. 425–448, 2017.

[64] M. Ehteram, H. Karami, and S. Farzin, “Reducing Irrigation Deficiencies Based Optimizing Model for Multi-Reservoir Systems Utilizing Spider Monkey Algorithm,” Water Resource Management, vol. 32, pp. 2315–2334, 2018.

[65] P. R. Singh, M. A. Elaziz and S. Xiong, “Modified Spider Monkey Optimization based on Nelder–Mead method for global optimization,” Expert Systems with Applications,vol. 110, pp. 264–289, 2018.

[66] S. Kumar, V. Kumar Sharma, and R. Kumari, “Self-adaptive spider monkey optimization algorithm for engineering optimization problems,” International Journal of Information, Communication and Computing Technology, vol. 2, no. 2, pp. 96–107, 2014.

[67] S. Kumar, V. K. Sharma, and R. Kumari, “Modified position update in Spider Monkey Optimization Algorithm,” International Journal of Emerging Technologies in Computational and Applied Science, vol. 7, no. 2, pp. 198-204, 2014.

[68] M. Dorigo, and L. M. Gambardella, “Ant colonies for the traveling salesman problem,” Biosystems, vol. 43, no. 2, 73–81, 1997.

Appendix A

List of Publications from the Thesis

[1] Safial Islam Ayon, M. A. H. Akhand, S. A. Shahriyar, and N. Siddique, “Spider Monkey Optimization to Solve Traveling Salesman Problem,” 2nd International Conference on Electrical, Computer and Communication Engineering (ECCE), Bangladesh, February 7-9, 2019.