Entry No. 97

Project/Thesis No.: CSER-M-14-01

Particle Swarm Optimization with Partial Search to Solve Traveling Salesman Problem

By

Shahina Akter

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in Computer Science and Engineering

Khulna University of Engineering & Technology
Khulna 9203, Bangladesh

February 2014

Declaration

This is to certify that the Thesis work entitled "Particle Swarm Optimization with Partial Search to Solve Traveling Salesman Problem" has been carried out by Shahina Akter in the Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh. The above thesis work or any part of this work has not been submitted anywhere for the award of any degree or diploma.

[illegible]	Shahina Akter
Signature of Supervisor	Signature of Candidate
Dr. Muhammad Aminul Haque Akhand	**Shahina Akter**
Associate Professor	Roll: 0907503
Dept. of Computer Science and Engineering	Dept. of Computer Science and Engineering
Khulna University of Engineering & Technology	Khulna University of Engineering & Technology

Approval

This is to certify that the thesis work submitted by Shahina Akter entitled "Particle Swarm Optimization with Partial Search to Solve Traveling Salesman Problem" has been approved by the board of examiners for the partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Department of Computer and Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh in February, 2014.

Acknowledgements

At first, I praise and like to thank Almighty Allah for showering all his blessings on me whenever I needed. It is my great pleasure to express my indebtedness and deep sense of gratitude to my Supervisor Dr. Muhammad Aminul Haque Akhand, Associate Professor, Dept. of Computer Science and Engineering (CSE), Khulna University of Engineering & Technology (KUET) for his continuous encouragement, constant guidance and keen supervision throughout the course of this study.

I am extremely grateful to all the faculty members of the Dept. of CSE, KUET to have their privilege of intensive, in-depth interaction and suggestions for the successful completion of my master degree. I would like to acknowledge my sense of gratitude to Mr. Al-Mahmud, and Mr. Sharif Sazzadur Rahman for their kind help in programming and writing the thesis.

February, 2014 Author

Abstract

The traveling salesman problem (TSP) is well-known combinatorial optimization problem. TSP requires to find the shortest circular tour visiting every city exactly once from a set of given cities. TSP is the most famous 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 as a general test bench. Recently nature inspired population based methods including PSO has drawn great attraction to solve TSP. In this thesis introduce a Particle Swarm Optimization (PSO) base algorithm to solve TSP in different way which is defined as Velocity Tentative Particle Swarm Optimization (VTPSO). Existing method introduced the idea of Swap Operator (SO) and Swap Sequence (SS) in PSO to handle TSP. In TSP, each particle represents a complete tour and velocity is measured as a SS consisting with several SOs. A SO indicates two positions in the tour that might be swapped. In the existing method, a new tour is considered after applying a complete SS with all its SOs. Whereas, every SO implantation on a particle (i.e., a solution or a tour) gives a new solution and there might be a chance to get a better tour with some of SOs instead of all the SOs. The objective of the study is to achieve better result introducing using such partial search option for solving TSP. The proposed Velocity Tentative Particle Swarm Optimization (VTPSO) algorithm is shown to produce optimal solution within a less number of generations than Self-Tentative PSO (STPSO) and Swap Sequence based PSO (SSPSO) in solving several benchmark TSP problems.

Contents

		Page
Title Page		i
Declaration		ii
Approval		iii
Acknowledgement		iv
Abstract		v
Contents		vi
List of Tables		viii
List of Figures		ix
Nomenclature		x
Chapter I	**Introduction**	**1**
	1.1 Particle Swarm Optimization and its Application	1
	1.2 Objectives of the Thesis	3
	1.3 Organization of the Thesis	3
Chapter II	**Literature Review**	**4**
	2.1 Traveling Salesman Problem (TSP)	4
	2.2 Particle Swarm Optimization (PSO)	6
	2.3 Particle Swarm Optimization to Solve TSP	8
	2.3.1 Swap Operator (SO)	8
	2.3.2 Swap Sequence (SS)	8
	2.3.3 The Basic Swap Sequence (BSS)	9
	2.3.4 Swap Sequence based PSO (SSPSO) to Solve TSP	9
	2.3.5 Self-Tentative PSO (STPSO) to Solve TSP	11
	2.3.6 Enhanced Self-Tentative (ESTPSO) to Solve TSP	14
	2.4 Limitations of existing approaches	15

Chapter III	**Velocity Tentative Particle Swarm Optimization to Solve TSP**	**17**
	3.1 Aspect of Partial Search Solving TSP	17
	3.1.1 Difference between Partial Search and Local Search	19
	3.1.2 Effect Partial Search in Tour Cost Calculation	19
	3.2 Velocity Tentative Particle Swarm Optimization to Solve TSP	20
	3.2.1 Comparison and Contrast of VTPSO with the Traditional Methods	23
Chapter IV	**Experimental Studies**	**24**
	4.1 Bench Mark Data and General experimental Methodology	24
	4.2 Experimental Results and Performance Comparison	25
	4.3 Experimental Analysis	29
	4.3.1 Variation Effect of the Number of Particles in the Population	29
	4.3.2 Effect of Population Size and Total Generation on Tour Cost and Required Time	31
Chapter V	**Conclusions**	**34**
	5.1 Achievements	34
	5.2 Perspectives	34
References		35
Publications Resulting from the Thesis		41

List of Tables

Table No	Caption of the Tables	Page
4.1	Comparison among SSPSO, ESTPSO and VTPSO on as is of average tour cost achieved and required time to solve benchmark TSPs. The results are the average of 20 independent runs	26
4.2	Comparison among SSPSO, STPSO and VTPSO on the basis of Minimum and Maximum tour cost achieved in the 20 independent runs to solve benchmark TSPs	28

List of Figures

Figure No	Caption of the Figure	Page
2.1	An optimal tour for a typical Traveling Salesman Problem (TSP)	5
2.2	Steps of PSO Algorithm	7
2.3	Flowchart of Basic PSO	7
2.4	Steps of Swap Sequence based PSO (SSPSO) Algorithm for TSP	10
2.5	Flowchart Swap Sequence based PSO (SSPSO) for TSP	11
2.6	Steps of Self-Tentative PSO (STPSO) to Solve TSP	12
2.7	Flowchart of Self-Tentative PSO (STPSO) for TSP	13
2.8	Steps of Enhanced Self-Tentative PSO (ESTPSO) to Solve TSP	15
2.9	Flowchart of Enhanced Self-Tentative PSO (ESTPSO) for TSP	16
3.1	Steps of Velocity Tentative PSO (VTPSO) Algorithm for TSP	21
3.2	Flowchart of Velocity Tentative PSO(VTPSO) for TSP	22
4.1	Velocity Swap Sequence size and time over generation	30
4.2	Variation effect of population size on tour cost and requires time	32
4.3	Variation effects of fixed no of generation on tour cost and require time	33

Nomenclature

SS	Swap Sequence
SOs	Swap Operators
PS	Partial Search
UM	Updating Method
ST	Self-Tentative
TAM	Total Accumulating Method
PSO	Particle Swarm Optimization
TSP	Traveling Salesman Problem
VTPSO	Velocity Tentative Particle Swarm Optimization
STPSO	Self-Tentative Particle Swarm Optimization
SSPSO	Swap Sequence based Particle Swarm Optimization
ESTPSO	Enhance Self-Tentative Particle Swarm Optimization
BSSPSO	Basic Swap Sequence based Particle Swarm Optimization

Chapter I
Introduction

Particle Swarm Optimization (PSO) is a population based optimization technique on the metaphor of social behavior of flocks of birds or schools of fishes and has found popularity in solving difficult optimization problems. Each particle adjusts its position in the search space from time to time according to the flying experience of its own and of its neighbors. PSO 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. PSO has also found as an efficient method to solve combinatorial problems such as Traveling Salesman Problem (TSP). In this thesis, given attention to solve TSP with existing PSO based methods and investigate Partial Search mechanism in its velocity operation and finally develop Velocity Tentative Particle Swarm Optimization (VTPSO) algorithm.

1.1 Particle Swarm Optimization and its Application

Particle Swarm Optimization (PSO) [1-3] is a population based optimization technique on the metaphor of social behavior of flocks of birds or schools of fishes. PSO is a simple model of social learning whose emergent behavior has found popularity in solving difficult optimization problems. It firstly generates a random initial population, the population contains a number of particles, each particle represents a potential solution of system, and each particle is represented by three indices: position, velocity and fitness. At every step, each particle calculates its new velocity considering its previous best position and the best one among all the particles in the population. Each particle then moves to a new position (i.e., search a new point) based on the calculated velocity. These processes of iteration continue until the stopping criterion is reached. PSO has been investigated on various continuous optimization (e.g., function optimization) [4] and combinatorial optimization (e.g., Traveling Salesman Problem) tasks [5-6] and found it as an effective method [1-4, 7-22].

PSO 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 function optimization domain, the position of the i-th particle is represented as

$X_i = \{x_{i1}, x_{i2}, x_{i3} \ldots, x_{iD}\}$ in the D-dimensional search space. At every step each particle changes position based on its velocity represented as $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. The velocity of a particle depends on its previous best position, $P_i = (p_{i1}, p_{i2} \ldots p_{iD})$. and the best one among all the particles in the population $G = (g_1, g_2 \ldots g_D)$ also known as global best solution [23]

PSO has also found as an efficient method 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 the minimum distance visiting all the assigned cities exactly for once. To solve TSP with PSO, a different consideration for the particle's position and velocity is required. Each PSO particle represents a complete tour to solve TSP and velocity is something to change a tour to a new tour.

A number of PSO based methods have been investigated in the recent years [23-30] and most of the methods use Swap Sequence (SS) for velocity operation [23, 31]. A Swap Sequence (SS) is a collection of several Swap Operators (SOs) and may define as $= (SO_1, SO_2, SO_3, \ldots, SO_n)$, where SO denotes Swap Operator [23, 31, 32]. A SO indicates two positions in the tour that might be swapped. All SOs of a SS are applied maintaining order on a particle and gives a new tour, i.e., a particle having new solution in the PSO [31, 32].

The first PSO [31] based method for TSP, basic SS based PSO (SSPSO), transformed the operations of PSO of function optimization to TSP. The SSPSO calculates velocity SS for each particle considering its previous best position and the best one among all the particles in the population. It does not conceive any additional operation in PSO. Conceiving the idea of SSPSO other algorithms to solve TSP are Self-Tentative PSO and Enhance Self Tentative PSO [33]. Self-Tentative PSO (STPSO) introduces tentative behavior in BSSPSO that tries to improve each particle placing a node in a different position. Enhance Self-Tentative PSO (ESTPSO) tries to improve individual particle with block of nodes adjustment in addition to individual node adjustment of STPSO [33].

1.2 Objectives of the Thesis

In the existing PSO based algorithms, the new tour of TSP is considered after applying all the SOs of a SS and no intermediate measure is considered. It is notable that every SO implantation results in a new tour; and therefore, there might be a chance to get a better tour with some of SOs instead of all the SOs. The objective of the study is to achieve better result introducing such partial search in the SS based PSO for solving TSP.

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

- Review existing Particle Swarm Optimization (PSO) based methods for solving Traveling Salesman Problem (TSP).
- Investigate Partial Search mechanism in velocity SS operation and develop Velocity Tentative Particle Swarm Optimization (VTPSO) that might be an efficient method in solving TSP.
- Compare performance of the proposed VTPSO with exiting SS based PSO methods in solving benchmark TSPs.

1.3 Organization of the Thesis

The main attraction of this thesis is to present new PSO based algorithm to solve TSP. The thesis has five chapters. An introduction to PSO and its applications to solve optimization tasks has been given in Chapter I. Chapter wise overviews of rest of the thesis are as follows. Chapter II is for literature review that includes brief description about TSP, PSO and previous related work to solve TSP by PSO based algorithms. The chapter also indentifies the legging of existing methods and gives motivation to develop of new method. Chapter III explains the proposed Velocity Tentative Particle Swarm Optimization (VTPSO) to solve TSP in detail. Chapter IV reports the experimental result of VTPSO and compares performance of the proposed VTPSO with exiting PSO methods to solve benchmark TSPs. Chapter V is for the conclusions of this thesis together with the outline of future directions of research opened by this work.

Chapter II
Literature Review

The Particle Swarm Optimization (PSO) is a population based search algorithm based on the simulation of the social behavior of birds, bees or a school of fishes. Each individual within the swarm is represented by a vector in multidimensional search space. This vector has also one associated vector which determines the next movement of the particle and is called the velocity vector. The PSO algorithm also determines how to update the velocity of a particle. Each particle updates its velocity based on current velocity and the best position it has explored so far and also based on the global best position explored by swarm. With the success to solve difficult continuous optimization problems, PSO has also found as an efficient method to solve combinatorial problems such as Traveling Salesman Problem (TSP). The TSP is an NP-hard in combinatorial optimization problem whose solution space is discrete. In this chapter, we will discuss about Traveling Salesman Problem, Basic PSO algorithm and some PSO based algorithms that solved TSP.

2.1 Traveling Salesman Problem (TSP)

The traveling salesman problem (TSP) is a well-known combinatorial optimization problem [30, 34]. TSP requires to find the shortest circular tour visiting every city exactly once from a set of given cities [28]. 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 though which he 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 $\pi: \{0, \cdots, N-1\} \rightarrow \{0, \cdots, N-1\}$, 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 [34]:

$$f(\pi, C) = \sum_{i=0}^{N-1} C_{\pi(i),\ \pi((i+1)\ mod\ N)}, \qquad (2.1)$$

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, paths 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 the 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 travelling salesmen, which included examples of tours. In the 1850s, Sir William Rowan Hamilton 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. 2.1 indicates an optimal tour for TSP with several typical cities marked in dot.

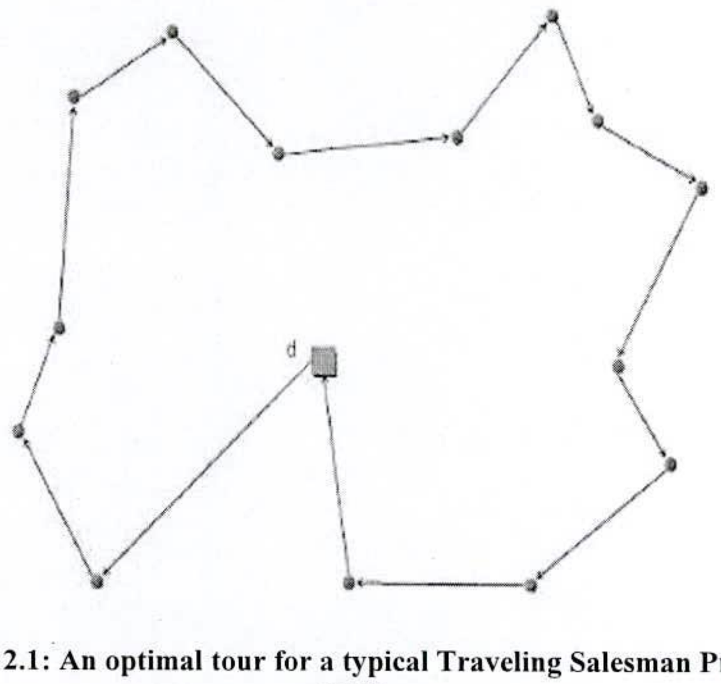


Figure 2.1: An optimal tour for a typical Traveling Salesman Problem (TSP).

TSP is a popular NP-hard problem and has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing, drilling a printed circuit board, computer wiring, order picking problem in warehouses, vehicle routing, X-Ray crystallography. In these applications, the concept *city* represents, for example, customers, soldering points, or DNA fragments, and the concept *distance* represents travelling times or cost, or a similarity measure between DNA fragments. 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 [3, 5, 6, 22, 34-49] as a general test bench. Recently nature inspired population based methods including PSO has drawn great attraction to solve TSP [23-33, 50-60].

2.2 Particle Swarm Optimization (PSO)

PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. PSO was originally designed and introduced by Eberhart and Kennedy [1]. Each particle's movement is influenced by its local best known position (P_i) and is also guided toward the best known positions in the search-space *(G)*, which are updated as better positions are found by other particles. Formally in PSO, each particle changes position $X_i = \{x_{i1}, x_{i2}, x_{i3} \dots, x_{iD}\}$ based on its calculated velocity $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$ that depends on its previous best position, $P_i = (p_{i1}, p_{i2} \cdots p_{iD})$ and the best one among all the particles in the population $G = (g_1, g_2 \cdots g_D)$. In each of iteration, each particle calculates its velocity according to the following formula.

$$V_i^{(t)} = \omega\, V_i^{(t-1)} + c_1 * r_1 \left(P_i^{(t-1)} - X_i^{(t-1)}\right) + c_2 * r_2\left(G^{(t-1)} - X_i^{(t-1)}\right) \qquad (2.2)$$

Where ω is inertia factor, c_1 and c_2 are learning factors, and r_1 and r_2 are vectors of random values between (0,1).

After calculating $V_i^{(t)}$ we can get the new position in next iteration by using the following formula:

$$X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)} \tag{2.3}$$

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

Step 2: Calculate the velocity and position of the particles according to Eq. 2.2 and 2.3.

Step 3: Update P_i if the new solution $X_i^{(t)}$ is superior to P_i .

Step 4: Update G if there is new solution $X_i^{(t)}$ is superior to G .

Step 5: Loop to **Step 2** until a termination criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

Step 6: Take the global best solution G as an outcome.

Figure 2.2: Steps of PSO Algorithm.

At every iteration step, for each particle, PSO calculates a new velocity value which I used to find the next position of particles. These processes of iteration continue until reaching stopping condition. Fig. 2.2 shows the steps of PSO algorithm and Fig. 2.3 is the flowchart of PSO algorithm.

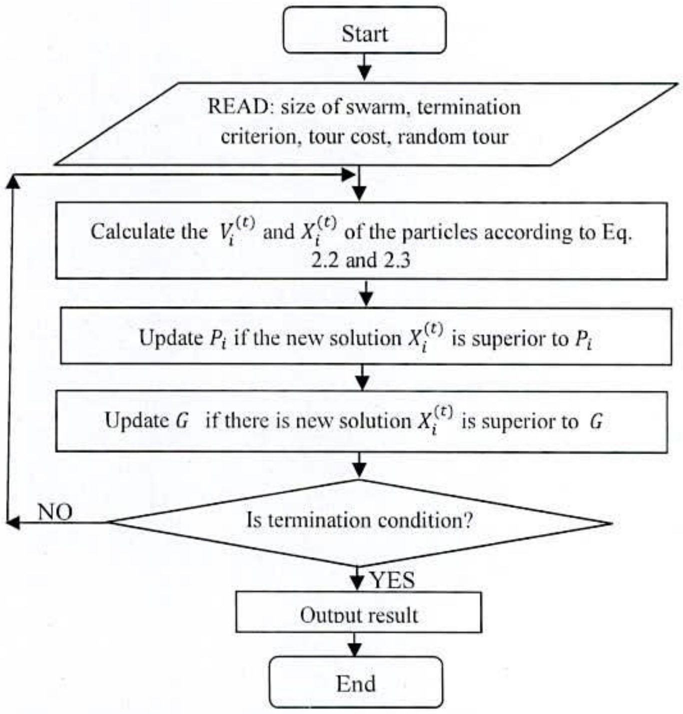

Start
READ: size of swarm, termination criterion, tour cost, random tour
Calculate the $V_i^{(t)}$ and $X_i^{(t)}$ of the particles according to Eq. 2.2 and 2.3
Update P_i if the new solution $X_i^{(t)}$ is superior to P_i
Update G if there is new solution $X_i^{(t)}$ is superior to G
NO
Is termination condition?
YES
Output result
End

Figure 2.3: Flowchart of Basic PSO Algorithm.

The basic PSO is successfully applied to function optimization problems. There are some other variations to the basic PSO that can be included to improve its performance such as Adaptive PSO [21], Hybrid PSO [12, 15, 16, 50, 51]. These variants of PSO have also several applications in various fields such as to solve traveling salesman problem [23-33, 50-60], solve time tabling problem [7, 20], improve power system reliability and security, system identification, intelligent control, planning, logistic, manufacture of microchips and so on [2, 8, 10- 19, 21, 60].

2.3 Particle Swarm Optimization to Solve TSP

TSP is a popular combinatorial optimization problem and a number of PSO based methods [23-33, 50-60] have been investigated for solving TSP up to date. A particle represents a tour and velocity is to change the tour towards a new tour when PSO [23, 25-33, 50-60] handle TSP. The methods use different techniques and parameters for calculating the velocity and then find new tour for particles. Among the methods, a number of prominent methods use the parameters Swap Operator (SO) and Swap Sequence (SS). Following subsections explain the operator SO and SS in detail and then explain the prominent methods.

2.3.1 Swap Operator (SO)

A Swap Operator (SO) contains a pair of indexes that indicates two cities may swap in a tour. Suppose, a TSP problem has 5 cities and a solution is $S = (1 - 3 - 5 - 2 - 4)$. Let a Swap Operator is $SO(2,4)$ then new solution with the following SO like below
$S' = S + SO(2,4) = (1 - 3 - 5 - 2 - 4) + SO(2,4) = (1 - 2 - 5 - 3 - 4)$,
here '+' means to apply SO on the solution S [31].

Swap Operator is the most important in solving TSP problem and its operation is similar to mutation operation of Genetic Algorithm.

2.3.2 Swap Sequence (SS)

A Swap Sequence (SS) is a collection of one or more Swap Operator(s) that might be applied on a solution one after another sequentially. To solve TSP, the velocity of PSO is represented as SS. The Swap Sequence can be defined as:

$$SS_{12} = (SO_1, SO_2, SO_3, \dots SO_n) \ , \tag{2.4}$$

where $SO_1, SO_2, SO_3, \ldots SO_n$ are Swap Operators. Swap Sequence acts on a solution applying all its *SOs* maintaining its sequence and then finally produces the new tour[31]. This can be described by the following formula:

$$S_2 = S_1 + SS_{12} = S_1 + (SO_1, SO_2, SO_3, \ldots, SO_n) \tag{2.5}$$

The order of *SOs* in a SS_{12} is important because implication of same SOs in different order may give different solutions from the original solution [31]. Considering Eq. 2.5 SS_{12} may get from solutions S_1 and S_2 in the following formula:

$$SS_{12} = S_2 - S_1 = (SO_1, SO_2, SO_3, \ldots, SO_n) \tag{2.6}$$

here '-' means need to apply *SOs* of SS_{12} on solution S_1 to get S_2.

As an example, if $S_1 = (1-2-3-4-5)$ and $S_2 = (2-3-1-5-4)$ then $SS_{12} = SO(1,3), SO(2,3), SO(4,5)$.

Moreover, several SSs may be merged into a new SS; the operator ⊗ defines as merging operation [31]. 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 SS_1 *and* SS_2 is

$$SS(new)=SS_1 \otimes SS_2 = \{SO(1,2), SO(5,2)\} \otimes \{SO(5,3), SO(4,1)\} \tag{2.7}$$
$$= SO(1,2), SO(5,2), SO(5,3), SO(4,1)$$

2.3.3 Basic Swap Sequence (BSS)

It is notable that different SSs acting on the same solution may produce the same new solution. All these SSs are named the equivalent set of SSs. In the equivalent set, the sequence which has the least SOs is called Basic Swap Sequence of the set or Basic Swap Sequence (BSS) in short.

As an example, both swap sequences $SS1_{12} = SO(1,3), SO(2,3), SO(4,5)$ and $SS2_{12} = SO(1,2), SO(2,1), SO(1,3), SO(2,3), SO(4,5)$ give same new solution $S_2 = (2-3-1-5-4)$ if applied on $S_1 = (1-2-3-4-5)$ individually [31]. Therefore $SS1_{12}$ is the Basic SS. It also find using Eq. 2.6 i.e., $S_2 - S_1$.

2.3.4 Swap Sequence based PSO (SSPSO) to Solve TSP

To solve TSP with PSO, the pioneer method [31] 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. For TSP, equations 2.2 and 2.3 are modified as follows:

$$V_i^{(t)} = V_i^{(t-1)} \otimes \alpha\left(P_i^{(t-1)} - X_i^{(t-1)}\right) \otimes \beta\left(G^{(t-1)} - X_i^{(t-1)}\right) \quad \alpha, \beta \epsilon [1,0]$$

$$V_i^{(t)} = V_i^{(t-1)} \otimes \alpha A \otimes \beta B \quad \alpha, \beta \epsilon [1,0] \tag{2.8}$$

$$X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)} \tag{2.9}$$

Where α, β are random number between 0 and 1, $\alpha\left(P_i^{(t-1)} - X_i^{(t-1)}\right)$ means all Swap Operators in Basic Swap Sequence $A = \left(P_i^{(t-1)} - X_i^{(t-1)}\right)$ should be maintained with the probability of α, it is the same as $B = \left(G^{(t-1)} - X_i^{(t-1)}\right)$. From here the bigger the value of α, the greater the influence of P_i, for more Swap Operators in A will be maintained, it is also the same as $\beta\ B$ [31]. For each particle, after calculating velocity SS using Eq. 2.8 each particle moves to a new tour solution ($X_i^{(t)}$) applying velocity SS on previous solution ($X_i^{(t-1)}$) using Eq. 2.9. Fig. 2.4 shows the steps of SSPSO algorithm to solve TSP and Fig. 2.5 shows the flowchart .

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 X_i in the swarm

a. Calculate velocity $V_i^{(t)}$ according to Eq. 2.8.
b. Update solution $X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)}$ using Eq. 2.9
c. Update P_i if the new solution $X_i^{(t)}$ is superior to P_i
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 2.4: Steps of Swap Sequence based PSO (SSPSO) Algorithm for TSP.

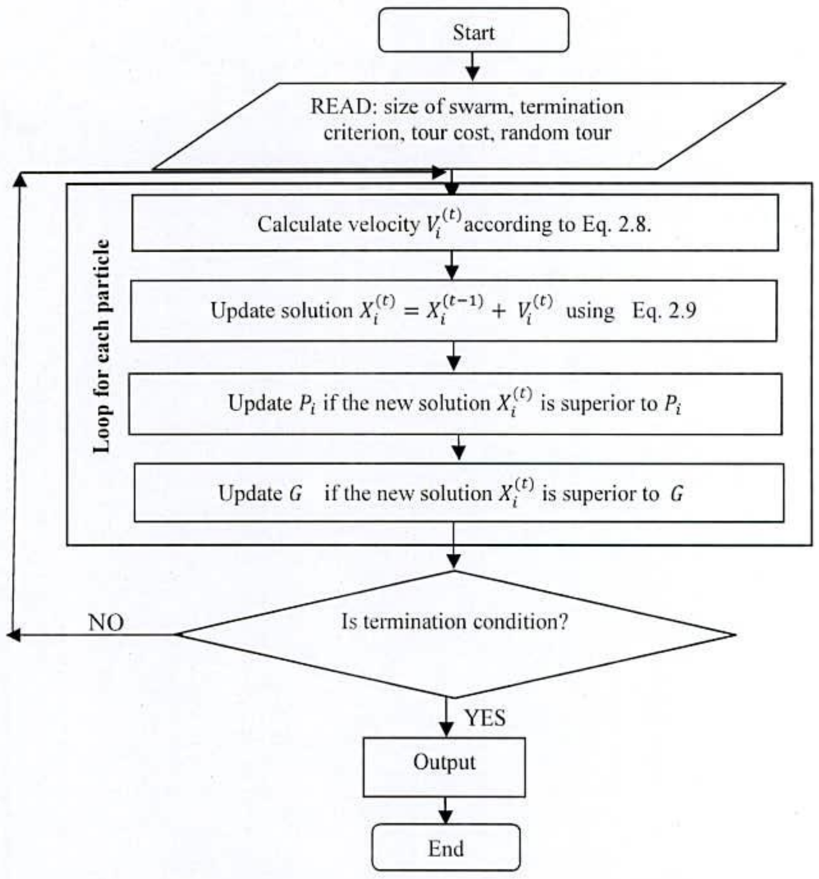

Start
READ: size of swarm, termination criterion, tour cost, random tour
Loop for each particle
Calculate velocity $V_i^{(t)}$ according to Eq. 2.8.
Update solution $X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)}$ using Eq. 2.9
Update P_i if the new solution $X_i^{(t)}$ is superior to P_i
Update G if the new solution $X_i^{(t)}$ is superior to G
Is termination condition?
NO
YES
Output
End

Figure 2.5: Flowchart Swap Sequence based PSO (SSPSO) for TSP.

2.3.5 Self-Tentative PSO (STPSO) to Solve TSP

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 man kind 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 [33].

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 the 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 can't 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 [33].

PSO calculates SS based velocity and change position of each particle using calculated velocity and action for tentative behavior is out of PSO operations. In each of iteration, each particle calculates its velocity according to the following formula.

$$V_i^{(t)} = \omega\, V_i^{(t-1)} \otimes c_1 . r_1 \left(P_i^{(t-1)} - X_i^{(t-1)}\right) \otimes\ c_2 . r_2 \left(G^{(t-1)} - X_i^{(t-1)}\right) \tag{2.10}$$

Where ω is inertia factor, c_1 and c_2 are learning factors, and r_1 and r_2 are vectors of random values between (0,1). Eq. 2.10 is more closer to original PSO equation Eq. 2.2 than Eq. 2.8 of SSPSO. After calculating $V_i^{(t)}$ the new position of a particle (i.e., tour for TSP) is calculated as like SSPSO by using the following formula:

$$X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)} \tag{2.11}$$

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 X_i in the swarm

a. Calculate velocity $V_i^{(t)}$according to Eq. 2.10
b. Update solution $X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)}$ using Eq. 2.11
c. Update P_i if the new solution $X_i^{(t)}$ is superior to P_i
d. Update G if the new solution $X_i^{(t)}$ is superior to G

Step 3: Tentative Operation on Each Particle X_i

a. Single Node Adjustment.
b. Update P_i and G if the new solution $X_i^{(t)}$ is superior to P_i and G respectively.

Step 4: Loop to **Step 2** until a termination criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

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

Figure 2.6: Steps of Self-Tentative PSO (STPSO) for TSP.

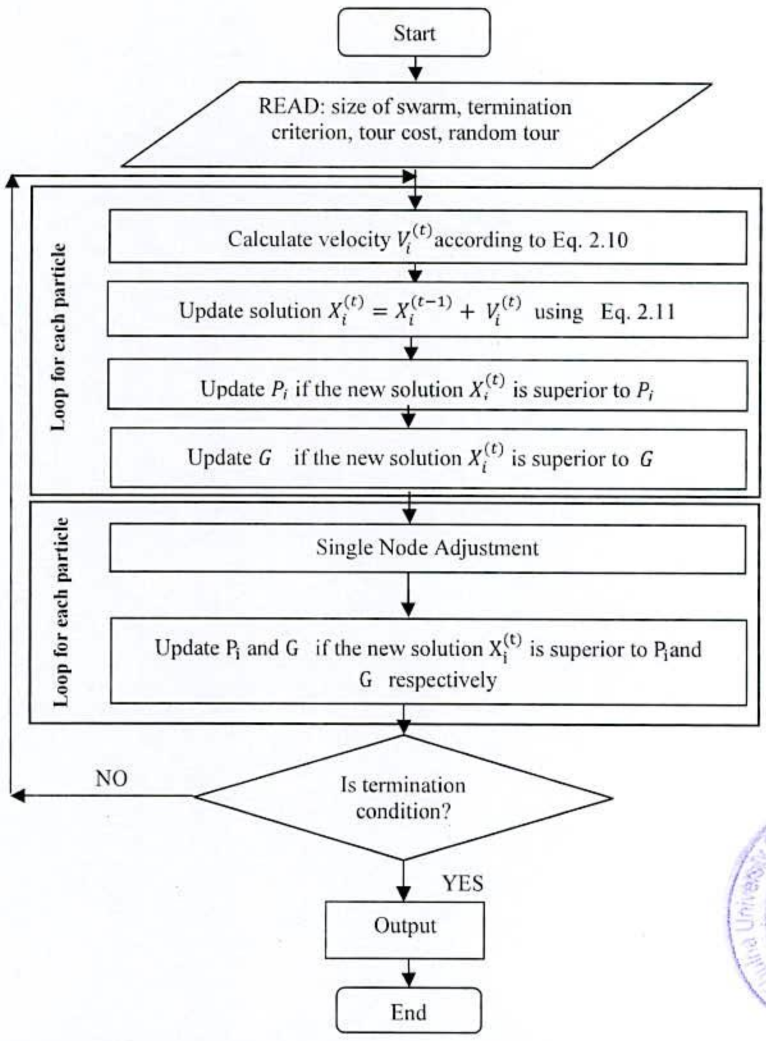

Start
READ: size of swarm, termination criterion, tour cost, random tour
Loop for each particle
Calculate velocity $V_i^{(t)}$ according to Eq. 2.10
Update solution $X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)}$ using Eq. 2.11
Update P_i if the new solution $X_i^{(t)}$ is superior to P_i
Update G if the new solution $X_i^{(t)}$ is superior to G
Loop for each particle
Single Node Adjustment
Update P_i and G if the new solution $X_i^{(t)}$ is superior to P_i and G respectively
NO
Is termination condition?
YES
Output
End

Figure 2.7: Flowchart of Self-Tentative PSO (STPSO) for TSP.

The steps of Self-Tentative PSO (STPSO) algorithm to solve the TSP are shown in Fig. 2.6 and Fig. 2.7 shows the flowchart. In STPSO steps the Step 3 is only the addition to SSPSO that try to improve individual particle with single node adjustment.

It is notable that single node adjustment might not be sufficient to get optimal result in some cases. It is hard to get the better position if the situations like this. For example, assume one particle's position is $X = (1-2-3-4-5-6-7-8-9-10)$ and the best one is $Y = (1-2-6-7-8-3-4-5-9-10)$, if there is no other better solution than X between them, it's hard to get Y from X using the single node adjustment algorithm [33].

2.3.6 Enhanced Self-Tentative (ESTPSO) to Solve TSP

Enhance Self-Tentative PSO (ESTPSO) introduces block node adjustment in STPSO [33]. ESTPSO tries to improve each particle placing a block of nodes in a different position after the single node adjustment of each particle. It may overcome limitation of single node adjustment but to select block length is difficult to determine. If block length is set from 2 to *N-2*, it is hard to end in the limited time. If the length is set to one constant, the adjustment process becomes stable and hard to find the better solution. Therefore, ESTPSO adopted a dynamic strategy based on generation. After the basic PSO operation and the single-node adjustment, the block size *k* is determined as a random number between 2 and K_{max}. And $K_{\max}$ changes according to the generation *t* and its calculation method are shown below.

IF ($N > 50$ and $t < 30\%$ of t_{max}) THEN $K_{max} = \lceil (1 / 10)\ N \rceil$

ELSEIF ($t > 65\%$ of t_{max}) THEN $K_{max} = \lceil (1 / 3)\ N \rceil$

ELSE $K_{max} = \lceil (1/5)\ N \rceil$

In this way, the subsequence max length (K_{max}) becomes longer with the *t* becomes bigger, it just comply with evolution process. Because at the beginning time each particle's position is random, and it's hard to find a better one if the adjustment length of the subsequence is longer. Secondly, for particle *i*, from the second node to $N - k + 1$, each subsequence as a whole, which length is *k*, should be done the actions same like the single node adjustment.

For the above example, particle's position is $X = (1-2-3-4-5-6-7-8-9-10)$, assume $k = 3$, then the first subsequence is (2,3,4), delete it and record the deleted position in the pointer $P = 2$, get $(1-5-6-7-8-9-10)$, trying insert (not really) it to other position, for example after the node 5 and get a new solution $(1-5-2-3-4-6-7-8-9-10)$, if the new solution is better than *X*, record the new position $P = 3$, else *P* didn't change. Then try other positions, and so on, until after the node 10, and then insert the subsequence to the position *P*, if there is no solution better than *X*, that is $P = 2$, *X* can be gotten again. The second subsequence is $(3-4-5)$, and the last subsequence is $(8-9-10)$, doing the same actions above. If the best solution is $Y = (1-2-6-7-8-3-4-5-9-10)$, then in the second time, Y must be getting after trying to insert the subsequence $(3-4-5)$ after the node 8. The steps of Enhanced Self-Tentative PSO (ESTPSO) algorithm to solve the TSP are shown in Fig. 2.8 and Fig. 2.9 shows the flowchart. In ESTPSO steps the block node adjustment (Step 3.b) after single node adjustment of Step 3.a is only the addition to STPSO as of STPSO. A modification on ESTPSO is also available that try to get better result trough reverse placement of a block is called Improved ESTPSO (IESTPSO) [33].

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 X_i in the swarm

a. Calculate velocity $V_i^{(t)}$ according to Eq. 2.10
b. Update solution $X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)}$ using Eq. 2.11
c. Update P_i if the new solution $X_i^{(t)}$ is superior to P_i
d. Update G if the new solution $X_i^{(t)}$ is superior to G

Step 3: Tentative Operation on Each Particle X_i

a. Single Node Adjustment.
b. **Block Node Adjustment**
c. Update P_i and G if the new solution $X_i^{(t)}$ is superior to P_i and G respectively.

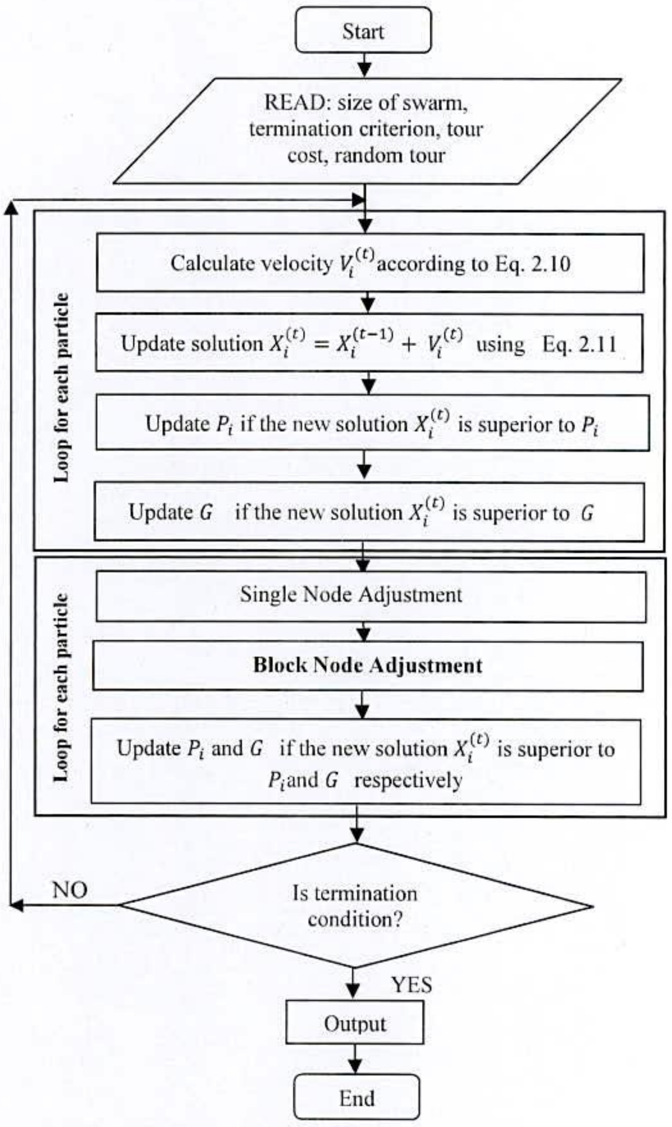
Step 4: Loop to **Step 2** until a termination criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

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

Figure 2.8: Steps of Enhanced Self-Tentative PSO (ESTPSO) to Solve TSP.

2.4 Limitations of existing approaches

The velocity for solving TSP in the above discussed PSO based methods (i.e., SSPSO, STPSO and ESTPSO) is the Swap Sequence (SS) with several Swap Operators (SOs). All SOs of a SS are applied maintaining order on a particle and gives a new tour i.e., a particle having new solution in the PSO. In the existing methods, the new tour is considered after applying all the SOs of a SS and no intermediate measure is considered. It is notable that every SO implementation gives a new tour, and therefore, there might be a chance to get a better tour (having better tour cost) with some of SOs instead of all the SOs. The objective of the study is to achieve better result introducing such partial search in the SS based velocity implication in PSO for solving TSP.



Start
READ: size of swarm, termination criterion, tour cost, random tour
Loop for each particle
Calculate velocity $V_i^{(t)}$ according to Eq. 2.10
Update solution $X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)}$ using Eq. 2.11
Update P_i if the new solution $X_i^{(t)}$ is superior to P_i
Update G if the new solution $X_i^{(t)}$ is superior to G
Loop for each particle
Single Node Adjustment
Block Node Adjustment
Update P_i and G if the new solution $X_i^{(t)}$ is superior to P_i and G respectively
NO
Is termination condition?
YES
Output
End

Figure 2.9: Flowchart of Enhanced Self-Tentative PSO (ESTPSO) for TSP.

Chapter III

Velocity Tentative Particle Swarm Optimization to Solve TSP

This chapter explains proposed partial search based Velocity Tentative Particle Swarm Optimization (VTPSO) to solve TSP. Proposed VTPSO considers Swap Sequence (SS) as velocity and calculates as like conventional standard SS based PSO methods described in the previous chapter. But VTPSO conceives a measure (called partial search) to apply such velocity to change particles position (for TSP to get a new tour). It measures the performance of a new tour applying each of SO and final velocity is considered for which it gives better tour. Therefore, the final velocity may be a portion (from the beginning) of calculated tentative velocity SS. Moreover, VTPSO conceives a moderate self-tentative technique that might improve its performance. The chapter starts with aspect of partial search and then explains the proposed algorithm. It also explains why proposed method might be efficient with respect to the exiting methods.

3.1 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 consider as velocity that may apply on a solution (i.e., a tour) to get a new tour [23, 31, 32]. In the tradition 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 better tour from a SS applying SOs one by one [31].

In Partial Search (PS) technique, the intermediate tours are considered as tentative tours and final tour is the best tentative tour. Eq. 3.1 and Eq. 3.2 are for velocity measure and position update of particles in existing PSO based method [23, 31]. The PS technique modifies the velocity implementation of Eq. 3.2.

$$V_i^{(t)} = \omega V_i^{(t-1)} \otimes \alpha\left(P_i^{(t-1)} - X_i^{(t-1)}\right) \otimes \beta\left(G^{(t-1)} - X_i^{(t-1)}\right) \qquad \alpha, \beta \epsilon [1,0] \qquad (3.1)$$

$$X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)} \qquad (3.2)$$

Suppose $V_i^{(t)} = SO_1, SO_2, SO_3, \ldots SO_n$ then

$$X_i^{1(t)} = X_i^{(t-1)} + SO_1$$
$$X_i^{2(t)} = X_i^{1(t)} + SO_2 = X_i^{(t-1)} + SO_1 + SO_2$$
$$\dots\dots\dots\dots\dots\dots\dots\dots$$
$$X_i^{n(t)} = X_i^{n-1(t)} + SO_n$$

In the above cases $X_i^{1(t)}, X_i^{2(t)}, \dots\dots, X_i^{(t)}$ are the tentative tours and the final tour $X_i^{(t)}$ is a tentative tour having the minimum tour cost.

$$X_i^{(t)} = X_i^{j(t)} \tag{3.3}$$

where $X_i^{j(t)}$ belongs minimum tour cost among $X_i^{1(t)}, X_i^{2(t)}, \dots X_i^{j(t)} \dots X_i^{n(t)}$

Finally the velocity considered is

$$V_i^{(t)} = SO_1, SO_2, SO_3, \dots SO_j \quad 1<j<n$$

The final velocity may also get from new and previous position of the particle

$$V_i^{(t)} = X_i^{(t)} - X_i^{(t-1)} \tag{3.4}$$

The above scenario may describe clearly with an example. Consider a tour $X_i^{(t-1)}$ of 10 cities where city position is the number of the city.

$$X_i^{(t-1)} = 1-2-3-4-5-6-7-8-9-10$$

If velocity SS is $SS = SO(1,4),\ SO(2,5), SO(2,4)$ the tentative tours are:

$$X_i^{1(t)} = X_i^{(t-1)} + SO(1,4) = \mathbf{4}-2-3-\mathbf{1}-5-6-7-8-9-10$$
$$X_i^{2(t)} = X_i^{1(t)} + SO(2,5) = 4-\mathbf{5}-3-1-\mathbf{2}-6-7-8-9-10$$
$$X_i^{3(t)} = X_i^{2(t)} + SO(2,4) = 4-\mathbf{1}-3-\mathbf{5}-2-6-7-8-9-10$$

The implementation of the SS gives two intermediate tours (i.e., $X_i^{1(t)}$ and $X_i^{2(t)}$) and finally reached at $X_i^{3(t)}$. Traditional methods only consider the last one as the final tour (i.e., $X_i^{(t)} = X_i^{3(t)}$) without evaluating the intermediate tours. On the other hand, the PS evaluates the intermediate tours too since all are the complete tours and an intermediate one (here $X_i^{1(t)}$ or $X_i^{2(t)}$) might be better than the final one (i.e., $X_i^{3(t)}$). In PS technique all three tours (i.e., $X_i^{1(t)}$, $X_i^{2(t)}$ and $X_i^{3(t)}$) are considered as tentative tours and velocity SS is considered as the tentative velocity. The final tour in PS technique is the best one among the three. If tour cost of $X_i^{2(t)}$ is better (i.e., lower) than $X_i^{1(t)}$ and $X_i^{3(t)}$ then $X_i^{2(t)}$ is considered as the final tour (i.e., $X_i^{(t)} = X_i^{2(t)}$) and final velocity is $V_i^{(t)} = SO(1,4),\ SO(2,5)$ that contains first two SOs of the calculated velocity SS [31].

3.1.1 Difference between Partial Search and Local Search

Partial search is not similar to local search. Local search explores 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 (the *search space*) 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 move from the old solution point towards a destined solution point.

3.1.2 Effect of Partial Search in Tour Cost Calculation

Partial search might not increase computational cost. It seems that evaluating the intermediate tours might increase computational cost of PS technique because it evaluates all the tentative tours while traditional methods only evaluate last one. But the technique of tour cost calculation minimizes the gap. A tour cost is the sum of individual link costs. To evaluate a tour, if all the links' cost are accumulated PS technique will take much time, in general (*n-1*) times of a traditional method if velocity SS contains *n* SOs. But it does not require considering all the links to get cost of a new tour from a tour which cost is known because a SO only changes four links because it index two cities in the tour which may interchange positions. So implementation of a SO need to discard costs of fours links and add costs four new links that associate with the indexed cities before and after interchange, respectively. As an example, to apply *SO(1,4)* on $X_i^{(t-1)} = 1-2-3-4-5-6-7-8-9-10$ to get $X_i^{1(t)} = X_i^{(t-1)} + SO(1,4) = 4-2-3-1-5-6-7-8-9-10$ it needs to discard cost of 1-2, 10-1, 3-4 and 4-5; and need to add cost of 4-2, 10-4, 3-1 and 1-5.

It will be more understandable when comparison of tour cost calculation between (1) method updating cost of the previous tour for city interchange, called Updating Method (UM) and (2) method accumulating all the links' cost, called Total Accumulating Method (TAM), which does not require previous tour cost. Suppose, *t* is the required time to read a link cost and *n* is the number SOs in the velocity SS. In UM, the time to get a tentative tour applying a SO is

$T_{SO} = 8t,$

And time for velocity SS with *n* SOs implementation is

$T_{SS}(\mathrm{UM}) = n^* T_{SO} = 8nt.$ (3.5)

On the other hand, time requires in TAM, tour is calculated after each SO implementation, is

$T_{SS}(\mathrm{TAM}) = n^* tN = ntN.$ (3.6)

Since the number of cities in the benchmark TSP problems in TSPLIB [61] varies from 14 to 493 (i.e., N>>8 in most of the cases) then tour cost calculation in UM is much efficient than TAM.

On the other hand, when intermediate tour costs are not required in the traditional methods, the required time in both UM and TAM methods are as follows.

$T(\mathrm{UM}) = 8nt$ (3.7)

$T(\mathrm{TAM}) = tN$ (3.8)

It is remarkable from Eq. 3.7 and Eq. 3.8 that TAM is better than UM for problem having few cities but UM might be efficient with respect to TAM for large sized problems. Moreover, time to calculate tour cost using update method is same (i.e., $8nt$) for both proposed partial search technique and traditional methods as seen from Eq. 3.5 and Eq. 3.7. Finally, partial search technique is used in this study may explore better result evaluating intermediate tentative tours without increasing computational time.

3.2 Velocity Tentative Particle Swarm Optimization to Solve TSP

Figure 3.1 shows the steps of the proposed Velocity Tentative Particle Swarm Optimization (VTPSO) to solve TSP considering the partial search technique; and 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 1) 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, 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 position of each particle conceiving proposed partial search based velocity tentative behavior (Step 2 in Fig. 3.1). The velocity Swap Sequence (SS) that it calculates (Step 2.a) using Eq. 3.1 is same as traditional methods.

```
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 $V_i^{(t)}$ according to Eq. 3.1. // Similar to Conv. Methods
    b. Update Solution $X_i^{(t)}$ Partial Search manner using Eq. 3.3
    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 3.1: Steps of Velocity Tentative PSO (VTPSO) Algorithm for TSP

For the velocity calculation of a particle, VTPSO considers (i) Last applied velocity ($v^{(t-1)}$), (ii) Previous best (P_i) solution of the particle and (iii) Global best (G) solution of the swarm. The calculated velocity does not apply on a position of the particle to get the new position like a traditional method. But VTPSO considers the calculated velocity as tentative velocity as its name regards. With the tentative velocity SS a number of tentative tours is evaluated and particle moves to the best one among those. The velocity SS that gave the best tour considers previous velocity ($V_i^{(t-1)}$) of the next iteration of Eq. 3.1. Finding the best tour is considered as the partial search and is the main "attraction" of VTPSO; therefore, Step 2.b in Fig. 3.1 for the operation is marked in bold-faced. The fitness of every new position of a particle is checked with P_i. If X_i is found better than P_i VTPSO applies Self-Tentative operation on (X_i) owing to improve it furthermore and then updates P_i (Step 2.c).

The Self-Tentative (ST) operation of VTPSO (Step 2.c (i)) consists with Single Node Adjustment of STPSO/ESTPSO and a simplified version of ESTPSO Block Node Adjustment. The block size k is determined as a random number between 2 and K_{max} and the value of K_{max} is defined as $N/2$, i.e., half of total cities of the problem. Moreover, ST operation is applied on a particle (i.e., a solution $X_i^{(t)}$) when it is found superior to P_i instead of all the particles as of ESTPSO. The ST operation on the selected particles might be helpful to improve overall performance of VTPSO with a minimal time complexity. The ST operation on selected particles is logical because single best solution (e.g., Global best G) is considered as the outcome. After ST operation P_i is updated (Step 2.c(ii) of Fig. 3.1) with new solution X_i. The fitness of every new position (X_i) of a particle is checked with G and updates G if X_i is shown better than G (Step 2.d).

VTPSO checks termination criterion at the end of each iteration (in Step 3 of the Fig. 3.1) and terminates if 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. 3.1. Fig. 3.2 shows the flowchart of the VTPSO.

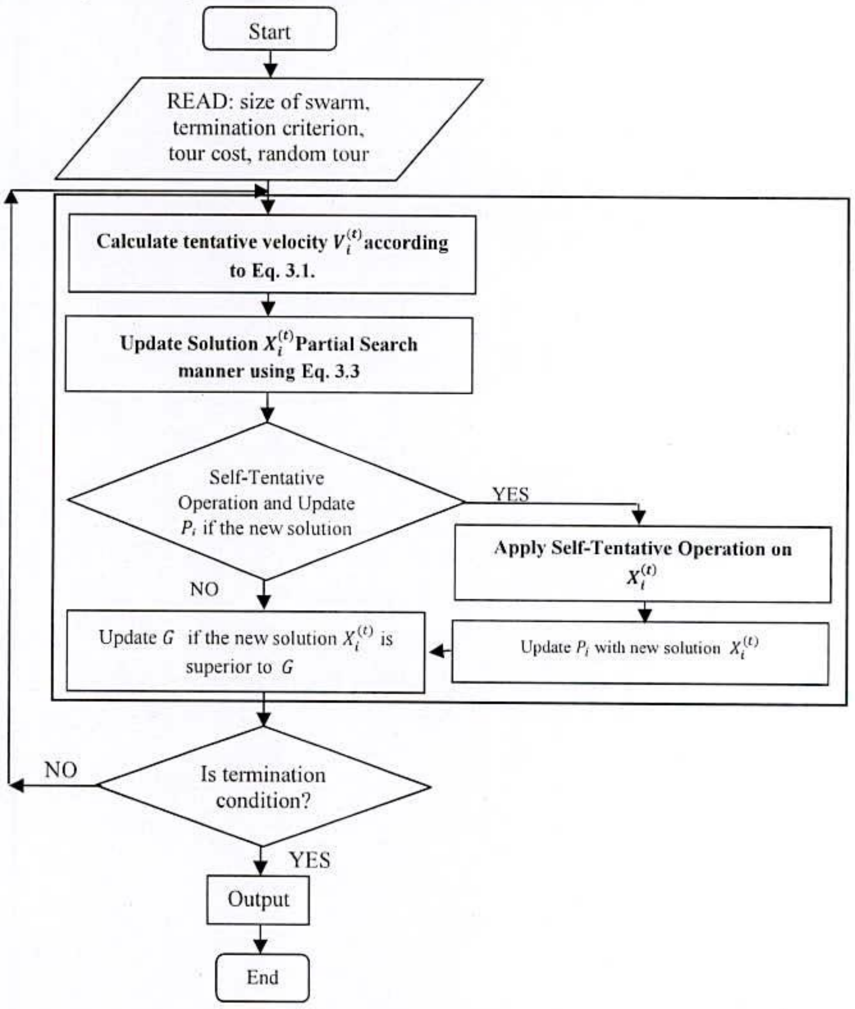

Start
READ: size of swarm, termination criterion, tour cost, random tour
Calculate tentative velocity $V_i^{(t)}$ according to Eq. 3.1.
Update Solution $X_i^{(t)}$ Partial Search manner using Eq. 3.3
Self-Tentative Operation and Update P_i if the new solution
YES
Apply Self-Tentative Operation on $X_i^{(t)}$
NO
Update G if the new solution $X_i^{(t)}$ is superior to G
Update P_i with new solution $X_i^{(t)}$
NO
Is termination condition?
YES
Output
End

Figure 3.2: Flowchart of Velocity Tentative PSO (VTPSO) for TSP

3.2.1 Comparison and Contrast of VTPSO with the Traditional Methods

Proposed Velocity Tentative PSO (VTPSO) introduced different way of getting new position with calculated velocity Swap Sequence (SS). VTPSO calculates velocity SS similar to conventional SS based methods (such as SSPSO, STPSO, ESTPSO), but in case of velocity implementation on a particle (i.e., current tour) VTPSO follows a different way. A conventional method gets new tour applying all the Swap Operators (SOs) [2,15,23] in the calculated velocity SS. On the other hand, VTPSO considers the calculated velocity as tentative velocity and evaluates a number of tentative tours applying SOs one after another sequentially. The final tour is considered as the best tentative tour and final velocity is also the part of SS from the beginning that gave the best tour. Although VTPSO evaluates a number of intermediate tours more than a conventional method it might not increase computational cost as explained in Section 3.1. On the other hand, VTPSO might converge faster than a conventional method because VTPSO may use yield instead a better tour with a part of SS than the tour with whole SS.

VTPSO conceives Self-Tentative behavior in order to achieve better outcome like Enhanced Self-Tentative (ESTPSO) but in a different way. After position update with PSO operation, both STPSO (described in Section 2.3.5) and ESTPSO (described in Section 2.3.6) try to improve every individual particle with single node adjustment. ESTPSO also follows a block node adjustment based Self-Tentative operation after single node adjustment. Thus, the tentative operation of STPSO and ESTPSO obviously add large computational cost with PSO operations since each and every particle is considered for tentative operation. On the other hand, VTPSO conceived Self-Tentative behavior of ESTPSO on the selected particles that might not increase computational cost much. Therefore, Self-Tentative operation might enhance performance of VTPSO with velocity tentative operation. Finally, VTPSO seems to be a cost effective method to solve TSP and return better tour. The coming chapter explains the effectiveness of the proposed VTPSO with experimental results to solve benchmark TSPs.

Chapter IV
Experimental Studies

This chapter experimentally investigates the efficacy of proposed Velocity Tentative Particle Swarm Optimization (VTPSO) algorithm to solve TSP. A set of benchmark problems were chosen as a test bed and the performance of VTPSO compared with two popular PSO based methods that are Swap Sequence based PSO (SSPSO) and Enhanced Self-Tentative PSO (ESTPSO). For fair comparison the experimental methodology were chosen carefully. Finally, an experimental analysis has been given for better understanding of the way of performance improvement in proposed method for solving TSP.

4.1 Bench Mark Data and General Experimental Methodology

In this study a suite of 30 benchmark problems are considered from TSPLIB [61] where number of cities varied from 14 to 200 and give diverse test bed. A numeric value in the problem name presents the number of cities in that tour. For example, *burma14* and *rat195* have 14 and 195 cities, respectively. A city is represented as a coordinate in a problem. Therefore the cost is found after calculating distance using the coordinates.

To investigate the proficiency of the proposed VTPSO, the study also implemented SSPSO and ESTPSO, and compared the experimental results among the three methods. The algorithms are implemented on Visual C++ of Visual Studio 2010. The experiments have been done on a Computer (Dell RM710, Intel (R) Xeon (R) CPU E5620 2.40 GHz, 16 GB RAM) with Windows Server 2008 Datacenter OS.

In the experimnets, the velocity inertia factor (i.e., ω) was considered as 1.0 and the values of learning factors (i.e., *c1* and *c2*) were taken randomly between (0, 1). For the fair comparison, the population size was 100; the number of generation was set at 500 as termination criteria for the algorithms. The selected parameters are not optimal values, but selected for simplicity as well as for fairness in observation.

4.2 Experimental Results and Performance Comparison

This section presents experimental results of the proposed Velocity Tentative Particle Swarm Optimization (VTPSO), Swap Sequence based PSO (SSPSO) and Enhanced Improved Enhanced Self-Tentative PSO (ESTPSO) and make a comparison among the results. SSPSO is the pioneer PSO based method to solve TSP and ESTPSO is the improved version of SSPSO and so far achieved good result. Table 4.1 compares tour cost achieved and required time to solve by the methods from 20 individual runs. Since experiments are conducted in a single machine with same experimental settings for all the methods, comparison of required time is a good choice to identify proficiency of a method. The bottom of the table shows the summary of the presented results.

The results presented in the Table 4.1 clearly indicate the effectiveness of the proposed VTPSO to solve TSP. The proposed method is shown the best method on the basis average tour cost of 30 problems. The average tour cost achieved by VTPSO is 21555.92; on the other hand SSPSO and ESTPSO achieved average tour costs of 160362.61 and 22246.28, respectively. Pair wise Win/Draw/Lose summary in the bottom of the table identified that VTPSO is better than SSPSO for all 30 problem individually. VTPSO is found better than ESTPSO for 21 cases out of 30 problems and preformed worse or equal for rest nine problems which are small sized problems such as *burma14* (problem with 14 cities) and *ulysses16* (problem with 16 cities).

Between two conventional methods, ESTPSO outperformed SSPSO for all the cases as it is observed from Table 4.1. As an example, SSPSO achieved tour cost of 1989.80 for eil76 problem, but ESTPSO achieved much better than SSPSO for the problem that is 583.93. Self-Tentative (ST) operation in ESTPSO is the element to get good result since such ST employment in ESTPSO is the main difference from SSPSO. But ESTPSO took on average 63.69 seconds to solve the same *eil76* problem which is much larger than SSPSO of 350.66 seconds. On average, ESTPSO took near about double time than SSPSO for most of the problems although SSPSO took larger time for few small sized problems such as *burma14*. That indicates ST operation is costly solution to improve performance.

Table 4.1: Comparison among SSPSO, ESTPSO and VTPSO on basis of average tour cost achieved and required time to solve benchmark TSPs. The results are the average of 20 independent runs.

Sl.	**Problem**	**Average Tour Cost (Standard Deviation)**			**Average Required Time in Millisecond**		
		SSPSO	**ESTPSO**	**VTPSO**	**SSPSO**	**ESTPSO**	**VTPSO**
1	burma14	*34.61* (1.16)	**30.87** (0.00)	**30.87** (0.00)	*47.68*	30.41	**24.39**
2	ulysses16	*83.94* (3.69)	**73.99** (0.00)	74.00 (0.01)	*56.78*	44.41	**28.12**
3	gr17	*3215.50* (260.48)	**2332.60** (0.00)	2342.80 (20.90)	*60.82*	39.23	**29.56**
4	ulysses22	*110.19* (4.59)	75.36 (0.09)	**75.35** (0.08)	*82.01*	57.93	**38.47**
5	gr24	*2295.50* (105.69)	**1249.80** (0.00)	**1249.80** (0.00)	*89.77*	51.87	**42.77**
6	fri26	*1229.10* (73.02)	**635.58** (0.00)	637.41 (4.94)	*98.74*	67.38	**45.85**
7	bays29	*17777.00* (655.35)	**9074.20** (0.00)	9092.90 (52.32)	*109.57*	82.99	**56.29**
8	eil51	*1218.60* (61.57)	**408.26** (4.86)	419.30 (5.76)	214.69	*273.49*	**117.35**
9	berlin52	*22046.00* (769.74)	**7750.70** (173.37)	7864.40 (192.15)	220.07	*295.32*	**125.21**
10	st70	*2831.80* (77.18)	**709.15** (17.70)	721.29 (19.43)	314.78	*467.49*	**188.13**
11	eil76	*1989.80* (54.33)	583.93 (11.87)	**574.67** (6.90)	350.66	*563.69*	**230.90**
12	gr96	*2693.10* (60.72)	561.68 (22.45)	**547.87** (16.43)	473.45	*777.36*	**316.78**
13	rat99	*6615.80* (186.85)	1402.30 (35.13)	**1337.80** (31.52)	493.56	*740.07*	**325.03**
14	kroa100	*132626.00* (3282.70)	23021.00 (1130.40)	**22436.00** (445.41)	461.56	*748.65*	**339.78**
15	rd100	*44747.00* (933.99)	8825.50 (265.82)	**8470.50** (162.87)	423.63	*673.56*	**318.30**
16	eil101	*2800.80* (51.55)	694.07 (13.45)	**678.21** (11.08)	503.45	*895.13*	**372.05**
17	lin105	*96405.00* (2576.40)	16249.00 (811.70)	**15805.00** (431.74)	540.92	*919.78*	**367.17**
18	pr124	*562623.00* (11019.00)	66905.00 (2164.90)	**63888.00** (1743.90)	668.14	*1226.20*	**490.94**
19	bier127	*534416.00* (9366.90)	128088.00 (3116.90)	**124534.00** (2346.30)	687.77	*1314.50*	**564.20**
20	ch130	*38976.00* (570.14)	6640.60 (135.48)	**6455.90** (132.63)	716.89	*1361.80*	**561.55**
21	pr136	*681844.00* (10731.00)	108154.00 (3747.30)	**104560.00** (2825.30)	757.34	*1444.40*	**584.41**
22	gr137	*4948.20* (128.08)	807.59 (21.75)	**775.87** (23.03)	765.93	*1437.20*	**557.98**
23	pr144	*679328.00* (11166.00)	67777.00 (1670.50)	**64511.00** (2389.40)	826.83	*1559.50*	**632.31**
24	ch150	*45170.00* (1101.10)	7321.00 (191.25)	**7045.50** (183.45)	874.67	*1610.00*	**728.94**
25	kroA150	*211527.00* (5224.30)	29333.00 (752.22)	**28611.00** (453.60)	872.64	*1485.20*	**726.30**
26	pr152	*875025.00* (17158.00)	78990.00 (2319.80)	**77184.00** (1903.80)	893.50	*1248.10*	**638.04**
27	u159	*375685.00* (4953.10)	47143.00 (2080.70)	**45617.00** (1288.40)	944.10	*1457.20*	**764.97**
28	rat195	*19201.00* (282.06)	2673.20 (35.38)	**2575.50** (45.33)	1188.00	*2119.40*	**1126.20**
29	d198	*154767.00* (3406.70)	17307.00 (230.09)	**16616.00** (274.63)	1173.70	*2061.10*	**1096.20**
30	kroA200	*288649.00* (4083.30)	32572.00 (775.18)	**31936.00** (506.54)	1124.00	*2227.50*	**1131.10**
	Average	**160362.61**	**22246.28**	**21555.62**	**534.52**	**909.36**	**418.97**
	Best/Worst	**0/30**	**9/0**	**21/0**	**0/7**	**0/23**	**30/0**

Pairwise Win/Draw/Lose Summary

Method	**on Tour Cost**			**on Required Time**		
	SSPSO	**ESTPSO**	**VTPSO**	**SSPSO**	**ESTPSO**	**VTPSO**
SSPSO	-	**30/0/0**	**30/0/0**	-	**0/0/30**	**30/0/0**
ESTPSO		-	**21/2/7**		-	**30/0/0**

The interesting observation from results of Table 4.1 is that VTPSO is much more time efficient with respect to SSPSO and ESTPSO but provide suitable solution with minimal tour costs. It is already mentioned that population size was 100 and the number of generation was set at 500 as termination criteria for each of SSPSO, ESTPSO and VTPSO for results presented in the Table 4.1. Since the algorithms are tested on the same machine with defined fair setting (unbiased to any one of those), the time to get solution is also a good measure to identify proficiency of a method that differs due to algorithmic matter. Moreover, finding better result with lesser time is more interesting that provides VTPSO. Partial search and moderate Self-Tentative operation are the additional operation in VTPSO with respect SSPSO. Both operations help to find better solution early and size of velocity SS reduces with generation that might be reason to take less time by VTPSO than SSPSO for same number of generations with fixed population size. On the other hand, Self-Tentative operation on each particle at each iteration make ESTPSO computationally much expensive than SSPSO in general. Self-Tentative operation on selected particles and velocity partial search makes VTPSO faster convergence and therefore return good result with less time. As example, VTPSO took 318.30 seconds for rd100 problem. For the same problem, SSPSO took 423.63 seconds and ESTPSO took more than double time of VTPSO i.e., 673.56 seconds. But VTPSO achieved best result for the problem with tour cost of 8470.50. In general, VTPSO took 80% time of SSPSO and less than half of ESTPSO. Finally, on the basis of tour cost and required time VTPSO is the best method.

Table 4.2 compares Minimum and Maximum tour costs among SSPSO, ESTPSO and VTPSO from 20 independent runs for which the average results presented in Table 4.1. For few small sized problems both ESTPSO and VTPSO achieved same minimum tour cost for all 20 independent runs. For the problems (e.g., *burma14, gr24*) maximum tour cost is same as minimum tour cost and Standard Deviation along with the average result in the Table 4.1 is 0.0. On the other hand, most of the cases, especially for large sized problems, VTPSO is shown better than ESTPSO showing lower (i.e., better) value on the basis of minimum and maximum tour costs. Out of 30 cases, VTPSO is shown better than ESTPSO for 15 and 20 cases on the basis of minimum and maximum values, respectively. On the other hand, both ESTPSO and VTPSO outperformed SSPSO for all 30 cases. Finally, on the basis of results of Table 4.2 SSPSO is the worst and VTPSO is the best.

Table 4.2: Comparison among SSPSO, STPSO and VTPSO on the basis of Minimum and Maximum tour cost achieved in the 20 independent runs to solve benchmark TSPs.

Sl.	Problem	Minimum Tour Cost (No of Times Min. Cost Achieved)			Maximum Tour Cost		
		SSPSO	**ESTPSO**	**VTPSO**	**SSPSO**	**ESTPSO**	**VTPSO**
1	burma14	*31.84* (1)	**30.87** (20)	**30.87** (20)	*37.20*	**30.87**	**30.87**
2	ulysses16	*77.36* (1)	**73.99** (20)	**73.99** (8)	*90.83*	**73.99**	74.00
3	gr17	*2772.53* (1)	**2332.58** (20)	**2332.58** (16)	*3600.13*	**2332.58**	2399.21
4	ulysses22	*99.88* (1)	**75.31** (15)	**75.31** (15)	*117.43*	**75.51**	**75.51**
5	gr24	*2021.14* (1)	**1249.82** (20)	**1249.82** (20)	*2442.40*	**1249.82**	**1249.82**
6	fri26	*1089.92* (1)	**635.58** (20)	**635.58** (17)	*1371.62*	**635.58**	656.30
7	bays29	*16092.24* (1)	**9074.15** (20)	**9074.15** (16)	*18535.93*	**9074.15**	9390.83
8	eil51	*1076.10* (1)	**403.19** (7)	408.47 (1)	*1315.28*	**416.52**	431.85
9	berlin52	*19629.98* (1)	**7544.37** (3)	**7544.37** (1)	*22758.80*	8189.69	**8140.00**
10	st70	*2587.43* (1)	**677.11** (1)	682.57 (1)	*2904.96*	**738.01**	748.27
11	eil76	*1785.90* (1)	**564.34** (1)	**564.53** (1)	*2041.14*	614.10	**591.53**
12	gr96	*2583.57* (1)	521.24 (1)	**518.83** (1)	*2772.27*	594.24	**584.30**
13	rat99	*6120.98* (1)	1306.32 (1)	**1283.23** (1)	*6875.10*	1475.45	**1396.22**
14	kroa100	*125296.17* (1)	**21307.44** (1)	21335.5 (1)	*138458.56*	25609.86	**23186.78**
15	rd100	*42307.19* (1)	8311.79 (1)	**8201.04** (1)	*46214.23*	9202.88	**8847.30**
16	eil101	*2704.97* (1)	674.58 (1)	**653.81** (1)	*2893.52*	728.74	**706.69**
17	lin105	*87155.78* (1)	**14855.59** (1)	**14855.59** (1)	*99386.61*	17456.15	**16607.62**
18	pr124	*534928.73* (1)	62870.01 (1)	**60726.57** (1)	*580861.71*	72217.81	**69760.96**
19	bier127	*512662.15* (1)	123984.40 (1)	**120237.56** (1)	*544356.11*	135696.90	**130742.53**
20	ch130	*37710.67* (1)	6395.06 (1)	**6297.31** (1)	*39745.47*	**6849.52**	6852.59
21	pr136	*655352.06* (1)	**99067.00** (1)	99447.33 (1)	*696218.21*	116870.40	**110281.53**
22	gr137	*4589.89* (1)	767.64 (1)	**716.40** (1)	*5085.17*	855.11	**803.98**
23	pr144	*653534.68* (1)	63075.73 (1)	**59900.08** (1)	*697861.41*	70246.12	**68097.54**
24	ch150	*42972.08* (1)	6894.51 (1)	**6718.71** (1)	*46525.76*	7563.81	**7459.16**
25	kroA150	*195681.34* (1)	27983.97 (1)	**27869.67** (1)	*218763.84*	30687.62	**29305.50**
26	pr152	*829997.63* (1)	**74490.45** (1)	74894.94 (1)	*897840.19*	83866.68	**82723.01**
27	u159	*366524.64* (1)	43979.37 (1)	**43750.50** (1)	*383569.10*	50808.34	**48236.22**
28	rat195	*18437.56* (1)	2600.18 (1)	**2507.77** (1)	*19611.62*	2739.74	**2661.76**
29	d198	*148410.07* (1)	16989.01 (1)	**16065.36** (1)	*159704.54*	17810.34	**17004.08**
30	kroA200	*279599.07* (1)	31577.63 (1)	**31080.45** (1)	*294971.95*	34165.11	**32994.25**
	Average	**153127.79**	**21010.44**	**20657.76**	**164564.37**	**23629.19**	**22734.67**
	Best/Worst	**0/30**	**15/0**	**24/0**	**0/30**	**10/0**	**23/0**

Pairwise Win/Draw/Lose Summary

Method	on Minimum Tour Cost			on Maximum Tour Cost		
	SSPSO	**ESTPSO**	**VTPSO**	**SSPSO**	**ESTPSO**	**VTPSO**
SSPSO	-	**30/0/0**	**30/0/0**	-	**30/0/0**	**30/0/0**
ESTPSO		-	**15/9/6**		-	**20/3/7**

4.3 Experimental Analysis

This section first investigates why VTPSO require less time than SSPSO and ESTPSO to get the solution on the basis of size of velocity Swap Sequence over generation. This section also investigates the effect of different parameters on the performance of SSPSO, ESTPSO and VTPSO. The size of swarm (i.e., number particles in the population) and the number of iterations were varied to observe their effect on the methods. Three problems with different size were selected for the analysis in this section; the problems are *eil76, gr99* and *eil101*.

4.3.1 Velocity Swap Sequence and Time over Generation

Figure 4.1 presents size of average velocity Swap Sequence (SS), i.e., average number of Swap Operators (SOs) in it, and time elapsed from the beginning for sample runs of three different problems. A total of 100 particles are trained for 500 generations for each of SSPSO, ESTPSO and VTPSO; SS in the figure is the average value of the particles at different generations. A velocity SS holds several SOs; operation of a SS is the collective operations of individual SOs of it. Therefore, a large velocity SS (having many SOs) requires more time to calculate as well as to implement for getting a new tour than a small one. A particle's position or solution (X_i) closer to particle best (P_i) and/or Global best (G) generates smaller velocity SS.

The velocity SS in the SSPSO (Eq. 2.8) is the accumulation of previous velocity (v^{t-1}) and a portion (that is selected based on parameter α and β) of calculated SS considering present solution (X_i), particle best (P_i) and Global best (G). Therefore, size of SS increases for any problem and reaches saturation level as it is seen in the Figure 4.1. ESTPSO tries to improve every individual particle through Self-Tentative operation and considers a portion of previous velocity (ref to Eq. 2.10) in the present velocity calculation. Both the things might be the reason to give smaller SS than SSPSO as it is seen the Fig. 4.1 for all three problems. But due to Self-Tentative operation on all the particles at each generation ESTPSO demands much time than SSPSO (as it is seen for all three problems in Figure 4.1) although size of velocity SS of ESTPSO is smaller than SSPSO.

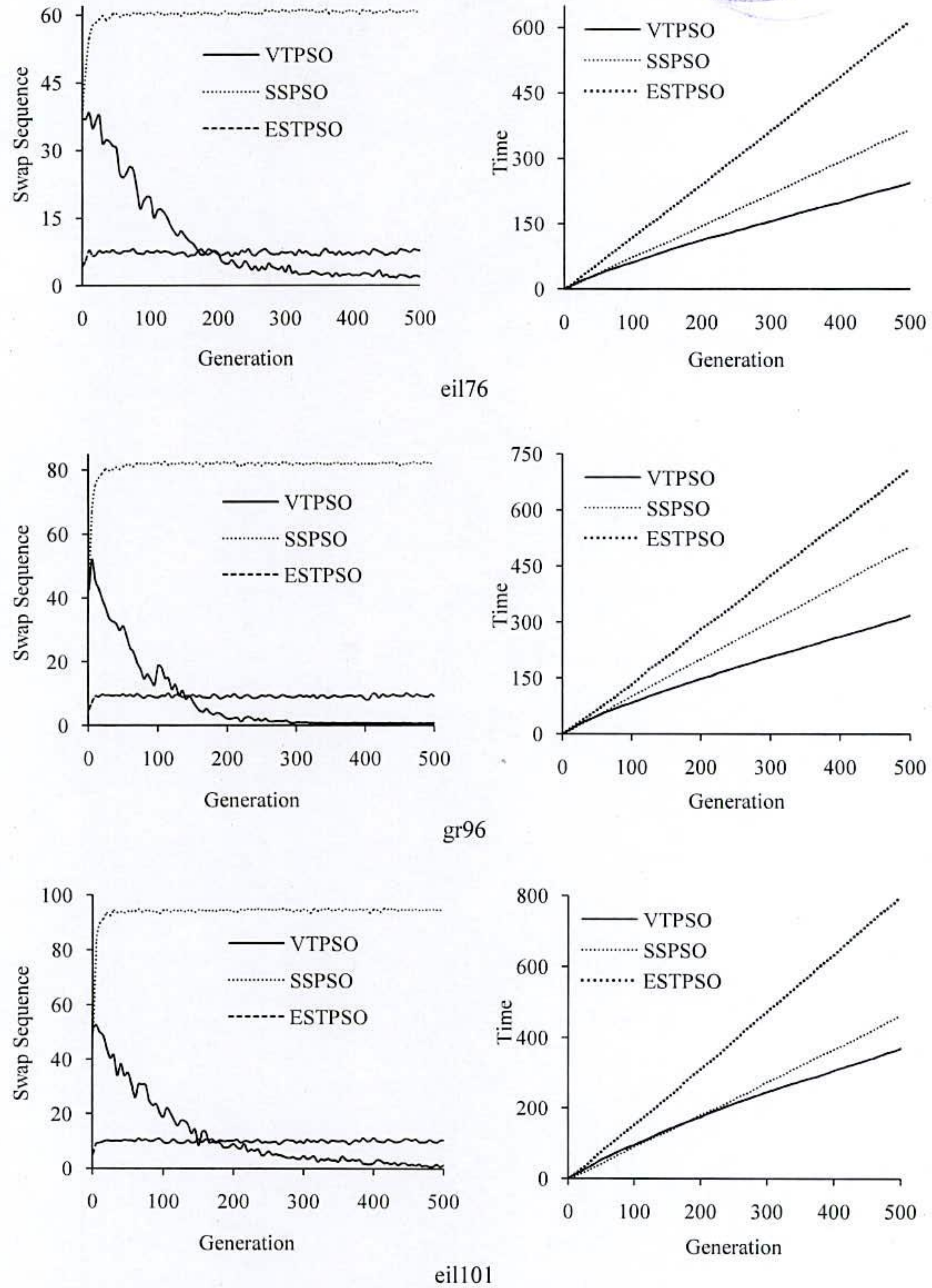


Figure 4.1: Velocity Swap Sequence size and time over generation.

VTPSO induce partial search in velocity SS implication and apply Self-Tentative operation on a particle when it is found better than particle best (P_i). Therefore, VTPSO seems slower than ESTPSO at early stage of generation when Self-Tentative operation is perform on most of the particles and later on (after 100 generations for *eil101*) it took less time than ESTPSO. On the other hand, the size of velocity SS for VTPSO decreases over generation, at the initial stage it is equal to SSPSO and after 100-200 generations it is less than ESPSO, in general. Finally, Partial Search and selected Self-Tentative operations make VTPSO faster convergence as well as time efficient.

4.3.2 Effect of Population Size and Total Generation on Tour Cost and Required Time

This section investigates performance of SSPSO, ESTPSO and VTPSO varying population size (i.e., number of particles in the swarm) and number of generation. The result presented in Tables 4.1 and 4.2 are for the fixed number of population size (=100) and generation (=500) for all the problems. It is interesting to observe how the algorithms perform on the variation of both the parameters. The experiments performed on the same machine explained before.

Figure 4.2 shows the achieved tour cost and required time for different population sizes from five to 500. The presented results are the average for ten independent runs. Since SSPSO perform is much worse result (having larger tour costs) than STPSO and VTPSO, its values are presented after scaling down for better visibility. It is seen from Fig. 4.2 that for very small population (e.g., 5) all three methods found to show worst tour cost and improve up to a certain population size. With tour cost improvement with population, VTPSO is shown better than ESTPSO for all any population size. Moreover, On the basis of required time, VTPSO is the best taking lowest time and ESTPSO is the worst taking largest time.

Figure 4.3 shows the achieved tour cost and required time for different fixed number of generations from 10 to 1000. The presented results are the average for ten independent runs. As like Fig. 4.2, the tour costs of SSPSO are presented in Fig. 4.3 after scaling down for better visibility. It is seen from the figure that all three methods are shown the worst tour costs for generation 10 and improve rapidly up to a certain value (e.g., 250 for *eil76*) and after that improvement is not significant. However, on the basis of achieved tour cost, SSPSO is the worst and VTPSO is the best for any value of generation. VTPSO is also the best on the basis of time requirement taking lowest time. Finally, the Figs. 4.2 and 4.3 ascertain VTPSO is a good method for solving TSP.

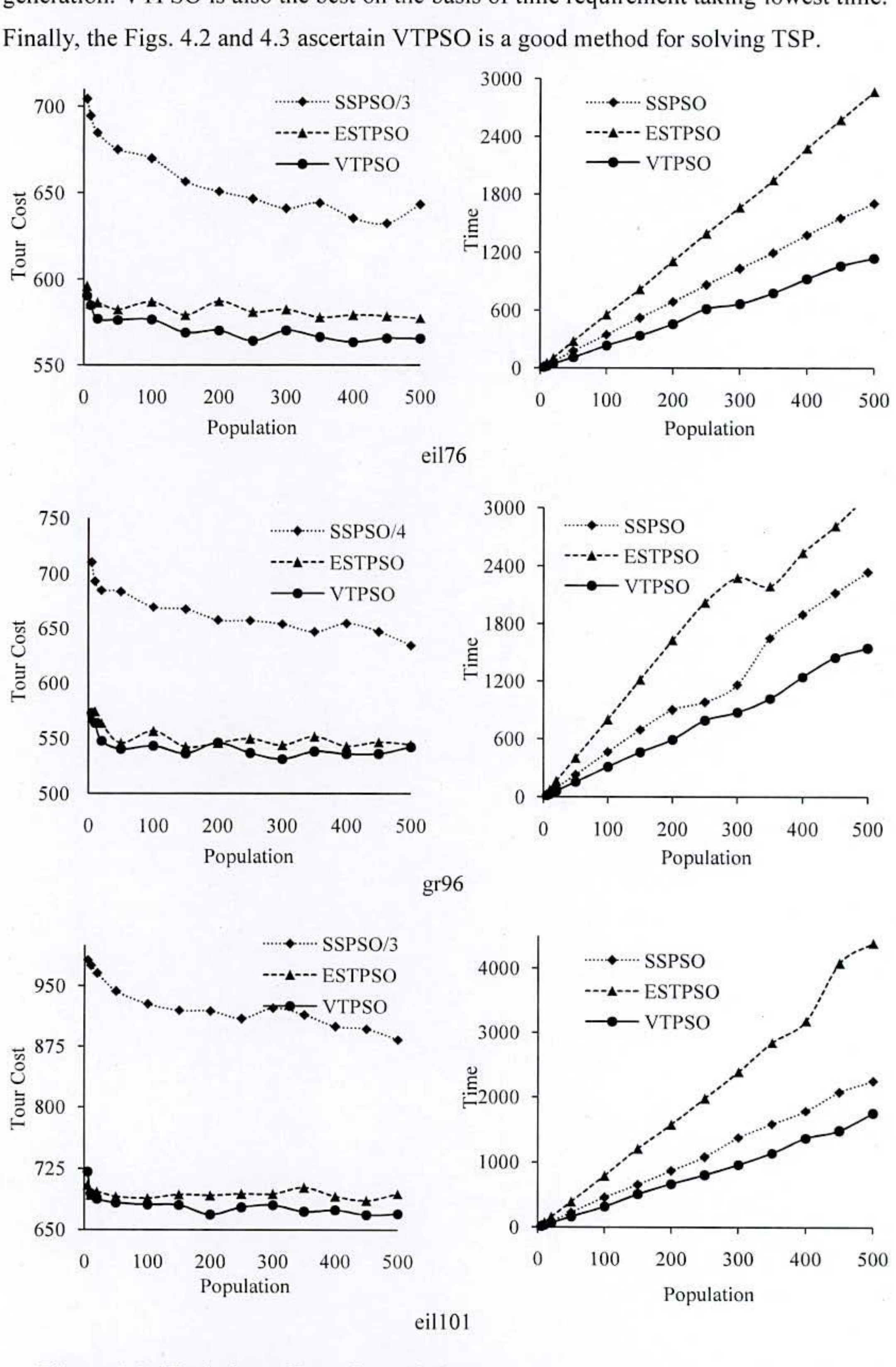
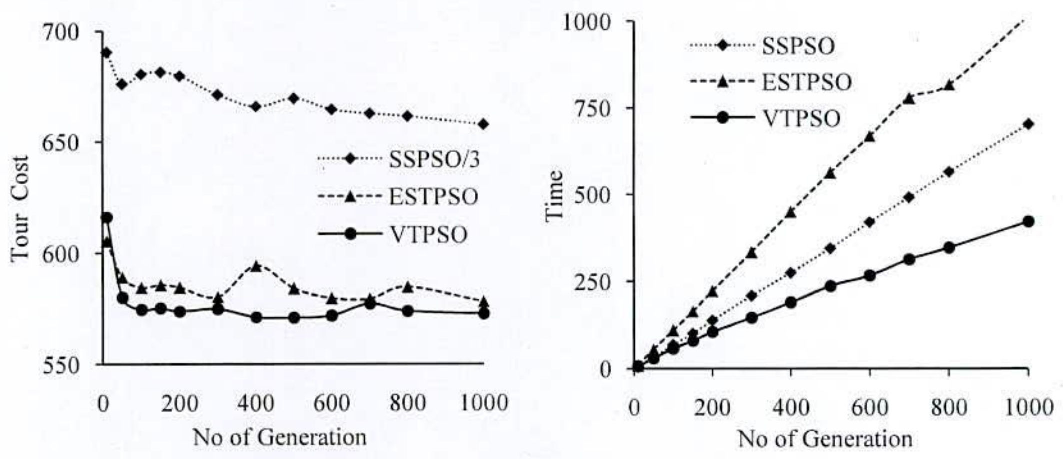
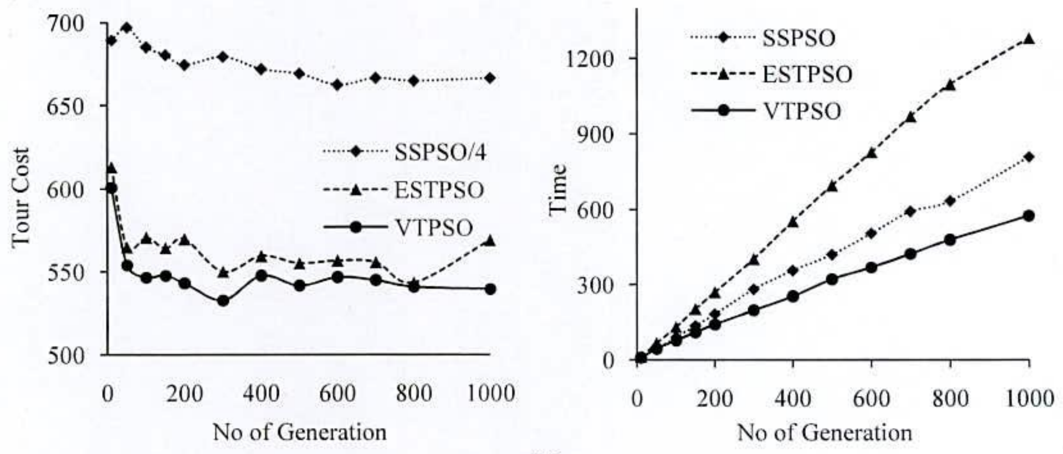


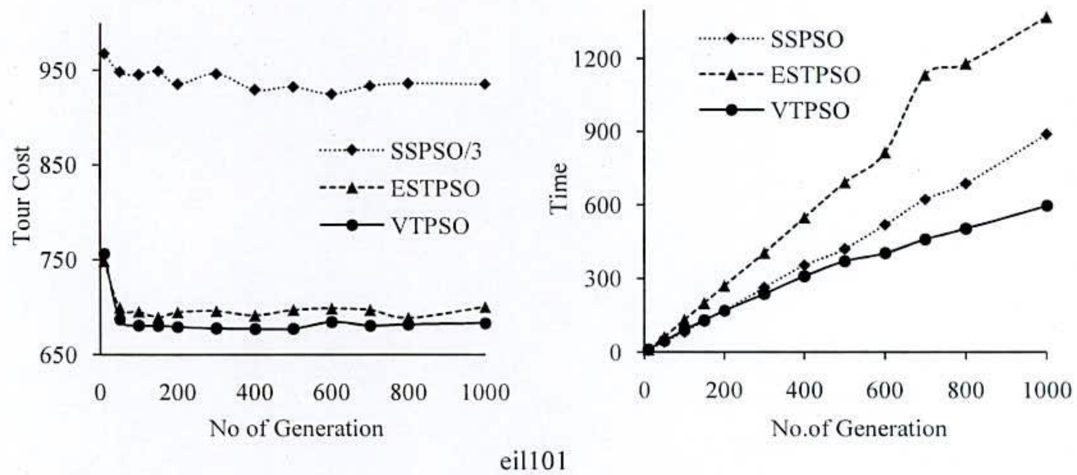
Figure 4.2: Variation effect of population size on tour cost and require time.



eil76



gr96



eil101

Figure 4.3: Variation effect of fixed no of generation on tour cost and require time.

Chapter V
Conclusions

Optimization has been an active area of research for several decades. Traveling Salesman Problem (TSP) is the most popular combinatorial optimization problem and interest grows in recent years to solve it new ways. Recently nature inspired population based methods had drawn great attraction to solve TSP. This thesis investigates a new Swap Operator (SO) based Particle Swarm Optimization (PSO) 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 Achievements

For TSP, each particle represents a complete tour and velocity is measured as a Swap Sequence (SS) consisting with several SOs. In the existing method, a new tour is considered after applying a complete SS with all its SOs of the velocity calculated. Since every SO implementation on a solution gives a new solution, we introduced a Partial Search and checked all the solutions for the calculated velocity SS to get best solution with velocity SS. The proposed Velocity Tentative PSO (VTPSO) is shown to produce optimal solution within a minimal time than conventional methods such SSPSO and Enhanced Self-Tentative PSO in solving benchmark TSPs. The reason behind the less time requirement is revealed from the experimental analysis is that VTPSO converge faster due to intermediate tour evaluation.

5.2 Perspectives

There are several future potential directions that follow from this study. This study considered partial search maintaining sequence of SOs in the velocity SS and indentifies that a portion of SS may give better than whole SS implementation. It is notable that SOs may be applied independently without sequence because velocity SS comes from three different sources: previous velocity, distance to previous best solution and distance to global best solution. Such consideration may give better result.

References

[1] R. Eberhart, J. Kennedy, "A New Optimizer Using Particles Swarm Theory," Roc Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan) IEEE Service Center, Piscataway,NJ:39-43, 1995.

[2] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," Journal of Information Processing Letters, Elsevier Publisher, pp. 317–325, September, 2003.

[3] 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.

[4] J. J. Liang, A. K. Qin, P. N. Suganthan and S Baskar , "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," Journal of evolutionary computation, vol. 10, issue. 3, pp. 281 – 295, May, 2006.

[5] 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.

[6] 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.

[7] D. C. Dang, R. N. Guibadj and A. Moukrim, "An effective PSO-inspired algorithm for the team orienteering problem," European Journal of Operational Research, Elsevier Publisher, vol. 229, p. 332–344, September, 2013.

[8] T. Geetha, B. Sowmiya, L. JayaKumar and A. S. Sukumar, "An Efficient Survey on Multi Colony- Particle Swarm Optimization (MC-PSO) Algorithm," International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, vol. 3, Issue 4, April, 2013.

[9] A. Yadav and K. Deep, "Shrinking hypersphere based trajectory of particles in PSO," Applied Mathematics and Computation, Elsevier Publisher, vol. 220, pp. 246–267, September, 2013.

[10] D. S. H. Ram, M. C. Bhuvaneswari, and S. S. Prabhu, "A Novel Framework for Applying Multiobjective GA and PSO Based Approaches for Simultaneous Area, Delay, and Power Optimization in High Level Synthesis of Datapaths," Journal of VLSI Design Vol.2012, Article ID 273276, p. 12, September, 2012.

[11] E. Montero, M. C. Riff and L. Altamirano, "A PSO algorithm to solve a Real Course+Exam Timetabling Problem," International conference on swarm intelligence, Cergy, France, 14-15 June, 2011.

[12] N. Talbi and K. Belarbi, "Fast Hybrid PSO and Tabu Search Approach for Optimization of a Fuzzy Controller," International Journal of Computer Science Issues, vol. 8, Issue 5, No 2, September, 2011.

[13] J. Langevel and A. P. Engelbrecht, "A Generic Set-Based Particle Swarm Optimization Algorithm," International conference on swarm intelligence, Cergy, France, June, 2011.

[14] Z. G. Zhou, "An Improved Ant Colony Optimization Supervised by PSO," Advanced Materials Research, vol. 108-111, pp.1354-1359, May, 2010.

[15] K. Premalatha and A. M. Natarajan, "Hybrid PSO and GA for Global Maximization," International Journal Of Open Problems Compt. Math, vol. 2, No. 4, December, 2009.

[16] Y. Zhang, M. Zhang, Y. C. Liang, "A hybrid ACO/PSO algorithm and its applications," International Journal of Modelling, Identification and Control, vol.8, No.4, p.309 – 316, 2009.

[17] A. Q. Mu, V. Caol and X. H. Wang, "A Modified Particle Swarm Optimization Algorithm," Journal of Scientific Research Open Access, Natural Science Publisher, Vol.1, No.2, pp. 151-155, August, 2009.

[18] W. T. Li, X. W. Shi and Y. Q. Hei, "An improved Particle Swarm Optimization algorithm for pattern synthesis of phased arrays," Journal of Progress In Electromagnetics Research, pp. 319–332, 2008.

[19] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. Part II: hybridization, combinatorial, multicriteria and constrained optimization, and indicative applications," Journal of Natural Computing, Springer Publisher, vol. 7, Issue. 1, pp.109-124, July, 2007.

[20] S. C. Chu, Y. T. Chen and J. H. Ho, "Timetable Scheduling Using Particle Swarm Optimization," International Conference on Innovative Computing, Information and Control, 2006.

[21] D. Chaojun and Q. Zulian, "Particle Swarm Optimization Algorithm Based on the Idea of Simulated Annealing," International Journal of Computer Science and Network Security, vol.6, No.10, October, 2006.

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

[23] M. A. H. Akhand, S. Akter, S. S. Rahman, 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.

[24] 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. 1694-0814, November, 2012.

[25] 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.

[26] 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, ISSN: 2160-0635, vol. 1, Number 1, September, 2011.

[27] A. J. Ouyang, Y. Q. Zhou, "An Improved PSO-ACO Algorithm for Solving Large-Scale TSP," Advanced Materials Research, vol. 143 – 144, p. 1154-1158, January, 2011.

[28] 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.

[29] 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.

[30] 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.

[31] K. P. Wang, L. Huang, C. G. Zhou, and W Pang, "Particle swarm optimization for traveling salesman problem," International Conference on Machine Learning and Cybernetics, Xi'an, pp. 1583–1585, November, 2003.

[32] 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, InTech Publisher, ISBN 978-953-307-053-7, pp. 318, February, 2010.

[33] Z. J. Wei, and S.W. Jian, "Improved Enhanced Self-Tentative PSO Algorithm for TSP," 6th International Conference on Natural Computation, Yantai, Shandong, pp. 2638-2641, 10-12 August, 2010.

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

[35] C. Filippi and E. Stevanato, "Approximation schemes for bi-objective combinatorial optimization and their application to the TSP with profits," The Journal of Computers & Operations Research, Elsevier Publisher, vol. 40, Issue 10, pp. 2418-2428, October 2013.

[36] B. Bollig and M. Capelle, "Priority functions for the approximation of the metric TSP," Journal of Information Processing Letters, Elsevier Publisher, vol. 113, issues 14–16, pp.584-591, July–August 2013.

[37] M. Mavrovouniotis and S. Yang, "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors," The Journal Of Applied Soft Computing, Elsevier Publisher, vol. 13, Issue 10, pp. 4023–4037, October, 2013.

[38] M. S. Kıran, H. Iscan and M. Gunduz, "The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem," International Journal Neural Computing & Applications, vol. 23, Issue 1, pp 9-21, July, 2013.

[39] M. Castro, K. Sorensen, P. Vansteenwegen and P. Goos, "A memetic algorithm for the travelling salesperson problem with hotel selection," Journal of Computers & Operations Research, Elsevier Publisher, vol. 40, Issue 7, pp. 1716–172, July, 2013.

[40] W. W. Guo and K. Tickle , "Solving the traveling salesman problem using cooperative genetic ant systems," The Journal Of Expert Systems with Applications, Elsevier Publisher ,vol. 39, Issue 5, pp. 5006–5011,April, 2012.

[41] C. M. Pintea, G. C. Crisan and M. Manea, "Parallel ACO with a Ring Neighborhood for Dynamic TSP," Journal of Information Technology Research, pp. 8, October, 2012.

[42] Kotzing ,T. Neumann, F. Roglin, H. Witt and Carsten, "Theoretical properties of two ACO approaches for the traveling salesman problem," International Conference on Swarm Intelligence, pp. 324-335, September, 2010.

[43] A. Ugur and D. Aydin, "An interactive simulation and analysis software for solving TSP using Ant Colony Optimization algorithms," Journal Of Advances in Engineering Software, Elsevier Publisher ,vol. 40, Issue 5, pp. 341–349, May, 2009.

[44] F. H. Khan, N Khan , S. Inayatullah and S. T. Nizami, "Solving TSP Problem by Using Genetic Algorithm," International Journal of Basic & Applied Sciences, vol. 9, pp. 79-88, December, 2009.

[45] X. F. Xie, and J. Liu, "Multiagent Optimization System for Solving the Traveling Salesman Problem (TSP)," IEEE Transactions on systems, man, and cybernetics-parts B: cybernetics, vol. 39, no.2, April, 2009.

[46] K. G. G. Daniel, "A Memetic Algorithm for the Generalized Traveling Problem," In the Proc. of International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO), Acireale, Sicily (Italy), 8-10 November, 2007.

[47] E. Ozcan and M. Erenturk, "A Brief Review of Memetic Algorithms for Solving Euclidean 2D Traveling Salesmen Problem," Department of Computer Engineering Yeditepe University Istanbul, 34755, Turkey, 2004.

[48] M. Dorigo and L. M. Gambardella, "Ant colonies for the traveling salesman problem," Accepted for publication in Bio Systems, May, 1997.

[49] L. V. Snydera and M. S. Daskin, "A random- key genetic algorithm for the generalized traveling salesman problem," European Journal of Operational Research, pp.38-53, May, 2005.

[50] Y. Marinakisa and M. Marinaki, "A Hybrid Multi-Swarm Particle Swarm Optimization algorithm for the Probabilistic Traveling Salesman Problem," Journal of Computers & Operations Research, Elsevier Publisher, pp. 432 – 442, March, 2010.

[51] M. Rosendo and A. Pozo, "A Hybrid Particle Swarm Optimization Algorithm for Combinatorial Optimization Problems," IEEE Congress the Conference on Evolutionary Computation, pp. 18-23, July, 2010.

[52] Q. Bai, "Analysis of Particle Swarm Optimization Algorithm," Journal of Computer And Information Science, Canadian Center of Science and Education Publisher, Vol. 3, No 1, pp. 180-184, February, 2010.

[53] H. Fan, "Discrete Particle Swarm Optimization for TSP based on Neighborhood," Journal of Computational Information Systems (JCIS), Binary Information Press Publisher, vol. 6, pp. 3407-3414, October, 2010.

[54] F. X. Xie and J. Liu, "Multiagent Optimization System for Solving the Traveling Salesman Problem (TSP)," IEEE Transactions on Systems, Man, Cybernetics, Part B Cybernetics, Vol 39, No. 2, April, 2009.

[55] M. A. Froushani and R. M. Yusuff, "Development of an Innovative Algorithm for the Traveling Salesman Problem (TSP)," European Journal of Scientific Research (EJSR), vol. 29 issue.3, pp. 349-359, 2009.

[56] M. R. Bonyadi, M. R. Azghadi and H. S. Hosseini, "Population-Based Optimization Algorithms for Solving the Travelling Salesman Problem," Travelling Salesman Problem, Book edited by: Federico Greco, InTech Publisher, Vienna, Austria, ISBN 978-953-7619-10-7, pp. 202, September, 2008.

[57] E. F. G. Goldbarg, M. C. Goldbarg and G. R. De Souza, "Particle Swarm Optimization Algorithm for the Traveling Salesman Problem," Travelling Salesman Problem, Book edited by: Federico Greco, ISBN 978-953-7619-10-7, pp. 202, 2008.

[58] M. F. Tasgetiren, P. N. Suganthan and Q. K. Pan, "A Discrete Particle Swarm Optimization Algorithm for the Generalized Traveling Salesman Problem," Genetic and Evolutionary Computation Conference, London, England, United Kingdom, 07 – 11 July, 2007.

[59] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and Q. X. Wang, "Particle swarm optimization-based algorithms for TSP and generalized TSP," Journal of Information Processing Letters, Elsevier Publisher, vol. 103, Issue. 5, pp. 169–176, March, 2007.

[60] L. Fang, P. Chen and S. Liu, "Particle Swarm Optimization with Simulated Annealing for TSP," International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, Corfu Island, Greece, 16-19 February, 2007.

[61] TSPLIB- A library of sample instances for the TSP. Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Publications Resulting from the Thesis

- M. A. H. Akhand, Shahina Akter, S. Sazzadur Rahman and M. M. Hafizur Rahman "Particle Swarm Optimization with Partial Search to Solve Traveling Salesman Problem," in Proc. of the International Conference on Computer and Communication Engineering (ICCCE 2012), Kuala Lumpur, Malaysia, pp. 118-121, July 3-5, 2012.

- M. A. H. Akhand, Shahina Akter, and M. A. Rashid "Velocity Tentative Particle Swarm Optimization to Solve TSP," in Proc. of International Conference on Electrical Information and Communication Technology (EICT2013), Khulna, Bangladesh, pp. 227-232, December 19-21, 2013.