Velocity Tentative PSO: An Optimal Velocity Implementation based Particle Swarm Optimization to Solve Traveling Salesman Problem

M. A. H. Akhand, *Member, IAENG,* Shahina Akter, M. A. Rashid and S. B. Yaakob

***Abstract*—This paper introduces an effective Particle Swarm Optimization (PSO) based algorithm for solving Traveling Salesman Problem (TSP). Among prominent PSO based methods, the proposed Velocity Tentative PSO (VTPSO) considers Swap Sequence (SS) for velocity operation of the particles. A velocity SS is a collection of several Swap Operators (SOs) where each one indicates two positions in a tour those might be swapped. The existing methods apply all the SOs of the calculated SS on a solution to get a new solution. Conversely, the proposed VTPSO considers the calculated SS as the tentative velocity and checks the tentative solutions when applies the SOs one after another sequentially. The best tentative tour with a portion of SS is considered as the next solution point of a particle in VTPSO. Such intermediate tentative tour evaluation not only helps to get better solution but also reduces overall computational time. The experimental results on a large number of benchmark TSPs reveal that the proposed VTPSO is able to produce better tour compared to other prominent existing methods.**

***Index Terms*—Swap Sequence, Partial Search, Particle Swarm Optimization, Traveling Salesman Problem.**

I. INTRODUCTION

PARTICLE Swarm Optimization (PSO) is a popular optimization method on metaphor of social behavior of flocks of birds or schools of fishes [1-3]. PSO is a simple model of social learning whose emergent behavior has been found popularity in solving difficult optimization problems. In PSO, each particle represents a potential solution and moves to a new position (i.e., search a new point) at every iteration based on the calculated velocity. The processes of iteration continue until the stopping criterion is reached.

PSO has been investigated on various continuous [2-6] and combinatorial optimization tasks [7-18]. PSO was proposed for continuous problems (e.g., function optimization) and has been proven to solve such problems effectively [2-3]. In function optimization domain, a particle represents a position in the multidimensional search space. At every step, each particle changes position based on its velocity depending on its previous best position and the best position particles ever visited.

PSO has also been found as an efficient method to solve combinatorial problems such as Traveling Salesman Problem (TSP) [7-16]. 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, each particle represents a complete tour as a feasible solution and velocity is a measure to update the tour for better solution. Many prominent PSO based methods use Swap Sequence (SS) for velocity operation [10-11]. A SS is a collection of several Swap Operators (SOs) and each one indicates two positions in a tour those might be swapped. All SOs of a SS are applied on a particle's tour maintaining order and hence implication of the SS transforms the TSP tour into a new one.

Basic SS based PSO (SSPSO) [10], the pioneer PSO based method for TSP, transformed PSO operations (i.e., velocity calculation and position update) of continuous domain to handle TSP, the combinatorial problem. SSPSO calculates the velocity SS for each particle considering its present tour, previous best tour and the global best tour. Conceiving the idea of SSPSO, other algorithms to solve TSP are Self-Tentative PSO (STPSO) and Enhanced Self-Tentative PSO (ESTPSO) [11-12]. STPSO considers tentative behavior that tries to improve each particle placing a node in a different position. ESTPSO also tries to improve each particle with block of nodes adjustment in addition to the single node adjustment of STPSO [11].

In this study, a new PSO based algorithm has been proposed and investigated for solving TSP. The proposed Velocity Tentative PSO (VTPSO) calculates velocity SS similar to existing methods but apply the SS in a different and optimal way. 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. On the other hand, VTPSO considers the calculated velocity SS as tentative velocity, measures tours with portions of it and conceive comparatively better new tour with a portion or full tentative SS. The proposed method is shown to perform well when tested on a suite of benchmark TSPs.

The rest of the paper is organized as follows. Section II describes some related PSO based methods those solve TSP. Section III explains the proposed VTPSO in detail. Section IV presents proficiency of the proposed method comparing with three prominent methods in solving benchmark TSPs. Finally, Section V concludes this paper with some remarks and outlines several future research directions opened by this study.

Manuscript received September 11, 2014; revised April 24, 2015. This work was supported in part by Khulna University of Engineering & Technology (KUET), Khulna, Bangladesh; University Sultan Zainal Abidin (UniSZA), Malaysia and University Malaysia Perlis (UniMAP), Malaysia.

M. A. H. Akhand is with the Department of Computer Science and Engineering, KUET (Corresponding author, phone: +880-41-774318, e-mail: akhand@cse.kuet.ac.bd, website: www.kuet.ac.bd/cse/akhand).

Shahina Akter is with the Department of Computer Science and Engineering, KUET (e-mail: shahina_akter23@yahoo.com).

M. A. Rashid is with Faculty of Design Arts and Engineering Technology, UniSZA (e-mail: marashid@unisza.edu.my).

S. B. Yaakob is with School of Electrical Systems Engineering, UniMAP (e-mail: shamshul@unimap.edu.my).

II. Existing PSO based Methods to Solve TSP

The TSP requires to find the shortest circular tour visiting every city exactly once from a set of given cities [8]. TSP is the most popular combinatorial problem because it has many real-world applications such as drilling a printed circuit board, computer wiring, order picking problem in warehouses, vehicle routing, X-Ray crystallography [19]. Almost every new approach for solving engineering and optimization problems has been tested on the TSP as a general test bench; and interest grows in the recent years to solve it new ways. Recently, a number of PSO based methods have been investigated for solving TSP [9-15]. Each particle holds a feasible tour as a solution point and its velocity conceives a measure to change the tour towards a new tour. The existing methods use different techniques and parameters for calculating the velocity and then find new tours for particles. A number of prominent PSO based methods use operators Swap Operator and Swap Sequence for velocity operation [10-12]. Following subsections explain the operators in detail and then present the prominent methods to make the paper self-contained.

A. Swap Operator (SO) and Swap Sequence (SS)

A SO [10-12] contains a pair of indexes that indicate two cities those might be swapped in a tour. Suppose, a TSP has five cities and a solution is $S = (1-3-5-2-4)$. Let a SO is *SO(2,4)*, then new solution (S') with it like below

$$S' = S + SO(2,4) = (1-3-5-2-4) + SO(2,4) = (1-2-5-3-4),$$

where '+' means to apply SO on the solution *S*.

A SS [10-12] is a collection of one or more SO(s) that might be applied on a solution one after another sequentially. To solve TSP, the velocity of PSO is represented as SS and a velocity SS can be defined as:

$$VSS = (SO_1, SO_2, SO_3, \dots SO_n), \quad (1)$$

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

$$S_2 = S_1 + VSS = S_1 + (SO_1, SO_2, SO_3, \dots, SO_n) \quad (2)$$

The order of SOs in the SS is important because implementation of same SOs in different order may give different solutions from the original solution. The *VSS* may also get from solutions S_1 and S_2 in the following equation.

$$VSS = S_2 - S_1 = (SO_1, SO_2, SO_3, \dots, SO_n), \quad (3)$$

where '-' means need to apply SOs of *VSS* 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 $VSS = SO(1,2), SO(2,3), SO(4,5)$.

Moreover, several SSs can be merged into a new SS; the operator $\otimes$ defines the 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 SS_1 and SS_2 is

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

It is noted that the 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 (BSS). As an example, both velocity swap sequences $SS_1 = SO(1,2), SO(2,3), SO(4,5)$ and $SS_2 = SO(1,2), SO(3,4), SO(3,5), SO(2,3), SO(2,4)$ give same new solution $S_2 = (2-3-1-5-4)$ if applied on $S_1 = (1-2-3-4-5)$ individually. Therefore SS_1 is the BSS. It also find using Eq. (3) i.e., $S_2 - S_1$.

B. Swap Sequence based PSO (SSPSO)

SSPSO [7] is the pioneer method to solve TSP that considers each particle as a complete tour and uses SS as velocity to get a new tour applying all its SOs on a tour. The SOs of the velocity SS of a particle is measured considering its previous best tour (P_i) and the best tour particles ever encountered (G). SSPSO follows Eq. (5) and Eq. (6) for velocity calculation and position update, respectively.

$$V_i^{(t)} = V_i^{(t-1)} \otimes \alpha \left(P_i - X_i^{(t-1)}\right) \otimes \beta \left(G - X_i^{(t-1)}\right) \ \alpha, \beta \epsilon [1,0] \quad (5)$$

$$X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)} \quad (6)$$

In Eq. (5), α, β are random numbers, $\alpha \left(P_i - X_i^{(t-1)}\right)$ means all SOs in BSS for $\left(P_i - X_i^{(t-1)}\right)$ should be maintained with the probability of α, which is the same for $\beta \left(G - X_i^{(t-1)}\right)$. The bigger the value of α (and β) the greater the influence of P_i (and G) will be maintained on present velocity calculation selecting more SOs from the portion. After velocity SS calculation using Eq. (5), each particle moves to a new tour solution ($X_i^{(t)}$) applying whole SS on its previous solution ($X_i^{(t-1)}$) using Eq. (6).

C. Self-Tentative and Enhanced Self-Tentative PSO

Self-Tentative PSO (STPSO) [11] applies a tentative behavior based operation after PSO operations to improve each particle in each iteration. For PSO operations, STPSO calculates velocity for each particle according to Eq. (7) and updates position like SSPSO using Eq. (6).

$$V_i^{(t)} = \omega V_i^{(t-1)} \otimes c_1 . r_1 \left(P_i - X_i^{(t-1)}\right) \otimes c_2 . r_2 \left(G - X_i^{(t-1)}\right) \quad (7)$$

In Eq. (7), c_1 and c_2 are learning factors, and r_1 and r_2 are random values between 0 and 1. The Eq. (7) is more closer to original PSO equation of function optimization considering a portion of previous velocity in present velocity based on the value of the scalling factor ω.

Tentative operation of STPSO tries to improve each particle placing a node of it in a different position in each iteration after PSO operations. For each particle, from the second node to the end the following actions are done: delete the node from the original position; measure fitness values with different positions and place it for which it gives the best fitness [11]. After this self-tentative operation, each particle would get a better position if any single node position changes can improve its fitness. This tentative behavior is important when random adjustment operators are hard to improve the solutions. But the single node adjustment is not sufficient to get optimal result in some cases [12]. To get better result, Enhanced Self-Tentative PSO (ESTPSO) considers block node adjustment in STPSO [11, 12].

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 block length selection is difficult to determine. If block length is set from 2 to *N-2* for a TSP having *N* cities, it is hard to end in the limited time. If the length is set to a

fixed value, the adjustment process becomes stable and hard to find the better solution. Therefore, ESTPSO adopted a dynamic strategy based on iteration. The block size k is determined as a random number between 2 and K_{max}. And K_{max} changes according to the iteration t and its calculation method is shown below.

IF ($N > 50$ and t< 30% of t_{max}) THEN $K_{max} = \lceil (1 / 10)\ N \rceil$
ELSE IF (t > 65% of t_{max}) THEN $K_{max} = \lceil (1 / 3)\ N \rceil$
ELSE $K_{max} = \lceil (1/5)\ N \rceil$ (8)

As of Eq. (8), the subsequence max length (K_{max}) becomes longer with t becomes bigger. ESTPSO then tries to improve each particle reallocating every subsequence block from the second node to $N - k + 1$ as like the single node adjustment. Detailed description regarding block node adjustment is available in the existing studies [11, 12]. ESTPSO is the best performed PSO based method so far for TSP.

Algorithm 1 shows steps of ESTPSO to solve TSP. In initialization (Step 1), it defines the number of particles and assigns a random TSP tour as well as a random velocity SS 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. ESTPSO checks termination criterion at the end of each iteration (in Step 4); 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, it continues updating positions of the particles again as indicates the loop to Step 2 from Step 4.

In ESTPSO, the block node adjustment (Step 3.b) after single node adjustment (Step 3.a) is only the addition to STPSO. On the other hand, employment of tentative operation (Step 3) in ESTPSO is the major addition to SSPSO. Moreover, ESTPSO uses Eq. (7) for velocity calculation, whereas SSPSO uses Eq. (5). A modification on ESTPSO is also available that also checks reverse placement of a block [12].

Algorithm 1: Enhanced Self-Tentative PSO (ESTPSO)

Step 1: Initialization

Step 2: For each particle X_i in the swarm
a. Calculate velocity $V_i^{(t)}$ according to Eq. (7)
b. Update solution using Eq. (6)
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 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 4: Loop to **Step 2** until a termination criterion is met

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

III. Proposed Velocity Tentative PSO (VTPSO)

This section explains proposed VTPSO to solve TSP. Similar to prominent PSO based methods as explained in the previous section, VTPSO considers SS for velocity operation of particles. 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 is a chance to get a better tour with some of SOs instead of all the SOs. The main objective of VTPSO is to achieve better result considering such intermediate tours.

Proposed VTPSO calculates velocity SS as like as the conventional methods that described already. But it conceives a measure called partial search (PS) to apply calculated SS to update particle's position (i.e., TSP tour). It measures performance of tours applying SOs of the calculated SS one after another, and the final velocity is considered for which it gives better tour. Therefore, the final velocity may be a portion (from the beginning) of calculated velocity SS. Moreover, VTPSO conceives a moderate self-tentative technique to improve its performance.

Algorithm 2 shows the steps of the proposed VTPSO to solve TSP considering the PS technique. Like other population based algorithm, VTPSO initializes the population with random solutions and tries to improve them at every iteration step. In initialization (Step 1), VTPSO defines the number of particles, assigns a random TSP tour and a random velocity SS 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 calculates velocity SS (Step 2.a) using Eq. (9) that is similar to traditional methods (e.g., ESTPSO), and considers (i) last applied velocity ($v^{(t-1)}$), (ii) previous best solution of the particle (P_i) and (iii) global best solution of the swarm (G). However, Eq. (9) of VTPSO for velocity SS calculation is simpler than Eq. (7) of ESTPSO. The Eq. (7) of ESTPSO requires values of three user defined parameters ω, c_1 and c_2; but VTPSO does not have any parameter to set and α, β in Eq. (9) are random numbers between 0 and 1.

$$V_i^{(t)} = V_i^{(t-1)} \otimes \alpha \left(P_i - X_i^{(t-1)}\right) \otimes \beta \left(G - X_i^{(t-1)}\right) \alpha, \beta \epsilon [1,0] \quad (9)$$

VTPSO does not apply calculated velocity SS on a particle to get its new position like a traditional method. But it considers the calculated velocity as tentative velocity as its name regards. A number of tentative tours is evaluated with the tentative velocity SS and the particle moves to the best one among those tentative tours. A portion or complete velocity SS that gave the best tour considers previous velocity ($V_i^{(t-1)}$) of the next iteration of Eq. (9). The best tour finding from several tentative tours is considered as the PS and is the

Algorithm 2: Velocity Tentative PSO (VTPSO)

Step 1: Initialization.

Step 2: For each particle X_i in the swarm
a. Calculate velocity $V_i^{(t)}$ using Eq. (9)
b. Update $X_i^{(t)}$ through Partial Search manner
c. Apply Tentative Operation on $X_i^{(t)}$ if is Superior to P_i
d. Update P_i if the new solution $X_i^{(t)}$ is superior to P_i
e. 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

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

main attraction of VTPSO; therefore, Step 2.b in Fig. 2 for such operation is marked in bold-faced.

PS seeks better result with a portion of calculated tentative velocity SS. It is already mentioned that a traditional method applies all the SOs of a velocity SS on a tour and generates a new tour with the whole SS although implementation of every successive SO transforms a tour into a new tour. While a traditional method ignores the intermediate tours, the PS technique explores the option of getting better tour considering the intermediate tours with a SS applying its SOs one by one.

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

$X_i^{1(t)} = X_i^{(t-1)} + SO_1$

$X_i^{2(t)} = X_i^{1(t)} + SO_2 = X_i^{(t-1)} + SO_1 + SO_2$

……

$X_i^{n(t)} = X_i^{n-1(t)} + SO_n$

In the above cases $X_i^{1(t)}$, $X_i^{2(t)}$, ….., $X_i^{n(t)}$ are the tentative intermediate tours; and the final tour $X_i^{(t)}$ in PS is the tentative tour having the minimum tour cost.

$$X_i^{(t)} = X_i^{j(t)}, \quad (10)$$

where $X_i^{j(t)}$ provides the minimum tour cost among $X_i^{1(t)}$, $X_i^{2(t)}$, … $X_i^{j(t)}$ … $X_i^{n(t)}$. Finally, the velocity considered as $V_i^{(t)} = SO_1, SO_2, SO_3, \ldots SO_j \quad 1 < j \leq n$.

The final velocity may also get from new and previous positions of the particle using the equation

$$V_i^{(t)} = X_i^{(t)} - X_i^{(t-1)}. \quad (11)$$

The above scenario can be described clearly with specific tour 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 $VSS = SO(1,4), SO(2,5), SO(2,4)$ the implementation of the SOs gives two intermediate tours (i.e., $X_i^{1(t)}$ and $X_i^{2(t)}$) and finally reaches at $X_i^{3(t)}$.

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

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

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

A traditional method only considers the last one to update a tour (i.e., $X_i^{(t)} = X_i^{3(t)}$). On the other hand, PS evaluates all the solutions since all three are the complete tours and an intermediate one (here $X_i^{1(t)}$ or $X_i^{2(t)}$) might be better than the last 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. The final tour in PS technique is the best one among the three. If tour $X_i^{2(t)}$ is found better than $X_i^{1(t)}$ and $X_i^{3(t)}$ then $X_i^{2(t)}$ is considered as the next solution point (i.e., $X_i^{(t)} = X_i^{2(t)}$). Applied velocity in this case contains first two SOs of the calculated velocity SS, i.e., $V_i^{(t)} = SO(1,4), SO(2,5)$.

VTPSO calculates fitness of every new position (X_i) of a particle and compares to its previous best P_i. If X_i is found better than P_i then VTPSO applies Self-Tentative (ST) operation on (X_i) owing to improve it furthermore. Such selective ST operation might be helpful to improve overall performance of VTPSO with a minimal time complexity. The ST operation of VTPSO (Step 2.c) consists of the Single Node Adjustment of STPSO/ESTPSO and a simplified version of ESTPSO Block Node Adjustment. The block size k is considered 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 given problem. After ST operation, P_i is updated (Step 2.d)) if new solution X_i is found better than P_i. The method also compares X_i with G and updates G accordingly if it is found inferior to X_i (Step 2.e).

VTPSO checks termination criterion at the end of each iteration (in Step 3); a sufficiently good fitness of G or a maximum number of iterations is considered as the termination criteria similar to other methods. If a termination criterion does not meet, it continues updating positions of the particles again as indicates the loop to Step 2 from Step 3.

A. Effect of Partial Search (PS) in Tour Cost Calculation

PS technique might not increase computational cost although it evaluates intermediate tours. The technique of tour cost calculation is an element of effectiveness of PS. In general, cost of a tour is calculated accumulating all individual links' costs. If all the links' cost are accumulated for every SO implementation PS will incur much time than a traditional method, normally (n-1) times for a velocity SS having n SOs. But it does not require to consider all the links to get cost of a new tour implementing SO(s) on a tour which cost is known. Since a SO indexes two cities in a tour to interchange their positions, implementation of a SO requires to discard costs of four links and add costs of four new links that associate with the indexed cities. Suppose a swap operator $SO(1,4)$ on $X_i^{(t-1)} = 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10$ gives $X_i^{(t)} = X_i^{(t-1)} + SO(1,4) = 4 - 2 - 3 - 1 - 5 - 6 - 7 - 8 - 9 - 10$. For the tour cost of $X_i^{(t)}$, link costs of 1-2, 10-1, 3-4 and 4-5 were discarded as well as link costs of 4-2, 10-4, 3-1 and 1-5 were added with the tour cost of $X_i^{(t-1)}$.

An analytical comparison can be outlined between two different tour cost calculation methods: (1) updating cost of the previous tour (that follows VTPSO) modifying only for the cities that a SO indicates to interchange; and (2) accumulating all the links' cost that uses a traditional method. Suppose, t is the required time to read a link cost and n is the number SOs in the velocity SS. In VTPSO, the time to get a tentative tour updating 8 links' cost for a SO is

$$T_{SO} = 8t. \quad (12)$$

Thus, total time requires in VTPSO to update a tour for a velocity SS having n SOs is

$$T_{\text{VTPSO}} = nT_{SO} = 8nt. \quad (13)$$

On the other hand, a traditional method calculates tour cost for a velocity SS accumulating all the links' costs of the tour. And for problem having N cities the time a method requires is

$$T = tN. \quad (14)$$

It is notable that tour cost calculation of VTPSO depends on the number of SOs, regardless of the number of cities in the problem. On the other hand, tour cost calculation of a traditional method depends on the number of cities in the problem. It is remarkable from Eq. (13) and Eq. (14) that a traditional method might be faster than VTPSO for problems

having few cities and VTPSO might compete for large sized problems. Moreover, VTPSO might be time efficient for small sized SS, i.e., a SS consists of less number of SOs. Finally, PS technique may explore better result evaluating intermediate tentative tours without increasing the computational time.

B. Comparison and Contrast with the Traditional Methods

Proposed VTPSO introduces different way of getting new tour with calculated velocity SS. It calculates velocity SS similar to conventional SS based methods (e.g., SSPSO, STPSO, ESTPSO), but in case of velocity implementation on a particle (i.e., current tour) it follows a different way. A conventional method gets new tour applying all the SOs 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 of the SS one after another sequentially. The next solution point (i.e., tour) is considered as the best tentative tour and applied velocity is a portion of SS from the beginning that gave the best tour. Although VTPSO evaluates a number of intermediate tours it might not increase computational cost as explained in the previous section. On the other hand, VTPSO might converge faster than a conventional method because it may conceive a better tour with a portion of SS than with the whole SS.

VTPSO conceives ST behavior owing to achieve better outcome like ESTPSO, but in a different manner. ESTPSO applies ST operation (i.e., single node adjustment and block node adjustment) on each and every individual particle after PSO operations (Step 3 of Fig. 1). Thus, ESTPSO obviously induces a huge computational cost with PSO operations. On the other hand, VTPSO applies ST operation on selected particles that might not increase computational cost much but might enhance the performance of VTPSO. Finally, VTPSO seems to be a cost effective method for better TSP solution.

IV. Experimental Studies

This section experimentally investigates the proficiency of the proposed VTPSO algorithm to solve benchmark TSPs. The performance of VTPSO compared to ESTPSO [9] (the prominent SS based PSO method) and other two prominent methods for TSP. The experimental methodology were chosen carefully for fair comparison. Finally, experimental analyses have also been given for better understanding of the way of performance improvement in the proposed method while solving TSP.

A. Bench Mark Data and Experimental Methodology

In this study, a suite of benchmark problems are considered from TSPLIB [20] where number of cities varied from 14 to 318 and give a 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 and therefore tour cost matrix is prepared after calculating distances using the coordinates.

For ESTPSO, the value of scalling factor ω of Eq. (7) is calculated using Eq. (15) that is identified to give better result according to the previous study [9].

$$\omega = 0.1 - (1 - sqr(1 - 2 * t/T)) * 0.05\,, \qquad (15)$$

where t and T are current iteration and total iteration, respectively. On the other hand, VTPSO does not require any parameter to set for velocity SS calculation as it uses Eq. (9) that is simpler than Eq. (7) of ESTPSO.

The algorithms were implemented on Visual C++ of Visual Studio 2010. For proper understanding, experiments have been conducted on a single machine (HP Pro, Intel (R) Core (TM) i5-3470 CPU 3.20 GHz, 4 GB RAM) with Windows 7 Professional OS.

B. Experimental Results Comparing with ESTPSO

This section presents experimental results of the proposed VTPSO and compares with the results of ESTPSO on a suite of 45 benchmark TSPs. For the fair comparison, the population size was 100; the number of iteration was set at 500 as the termination criteria for both VTPSO and ESTPSO. For each problem, outcomes of 20 individual runs by a method are summarized and considered in performance comparison. Since experiments were conducted in a single machine with same experimental settings for both the methods, comparison between the methods on the basis of required time to solve a problem is considered as a good choice to identify the proficiency of a method.

Table I compares problem wise achieved tour cost (average and best) and average required time between ESTPSO and VTPSO from 20 individual runs. Pair two tailed *t*-test was conducted to determine the significance in the variation of results. If tour cost of VTPSO was found significantly better than ESTPSO by *t*-test for a particular problem, it is marked with a plus (+) sign in the column of *t*-test evaluation. On the other hand, a minus (−) sign indicates VTPSO was significantly worse than ESTPSO for a particular problem. A single plus/minus means the tour cost difference between ESTPSO and VTPSO was statistically significant with 95% confidence interval and a double plus/minus is for 99% confidence interval. The bottom of the table shows the average outcome over all 45 problems.

The results presented in the Table I clearly indicate the effectiveness of the proposed VTPSO to solve benchmark TSPs. The proposed method is shown better than ESTPSO on the basis of average tour cost of 45 problems. The average tour cost achieved by VTPSO was 23241.30; on the other hand ESTPSO achieved average tour cost of 23969.16. VTPSO is found better than ESTPSO for 34 cases and the *t*-test shows that the performance of VTPSO is significantly better than ESTPSO on 33 problems. ESTPSO, however, significantly outperformed VTPSO on only two problems (i.e., fri26 and hk48). In general, ESTPSO was competitive to VTPSO for only small sized problems (e.g., burma14, ulysses16); otherwise VTPSO outperformed ESTPSO.

The interesting observation from results of Table I is that VTPSO is much more time efficient than ESTPSO but provides suitable solution with minimal tour costs. Since the algorithms were tested on same machine with defined fair setting (unbiased to any one of those), the time requirement differs due to algorithmic matter. Moreover, finding better result with lesser time is more interesting that is provided by VTPSO. PS in VTPSO encourages to find better solution early. In addition, velocity SS size might reduce with iteration in VTPSO and might enhanced it to take overall less time to solve a problem. On the other hand, ST operation on each

TABLE I
COMPARISON BETWEEN ESTPSO AND VTPSO OVER 20 INDEPENDENT RUNS TO SOLVE BENCHMARK TSPS.

Sl.	Problem	Average Tour Cost (Standard Deviation)		*t*-test eval. of VTPSO with ESTPSO	Minimum Tour Cost (No of Times Min. Cost Achieved)		Average Req. Time in Second	
		ESTPSO	VTPSO		ESTPSO	VTPSO	ESTPSO	VTPSO
1	burma14	30.87 (0.0)	30.87 (0.0)		30.87 (20)	30.87 (20)	16.96	**13.03**
2	ulysses16	**73.99** (0.0)	74 (0.02)		73.99 (18)	73.99 (5)	21.71	**15.48**
3	gr17	**2332.58** (0.0)	2337.2 (13.86)		2332.58 (20)	2332.58 (18)	19.15	**15.51**
4	gr21	**2672.27** (0.0)	2681.41 (27.42)		2672.27 (20)	2672.27 (18)	25.84	**19.34**
5	ulysses22	**75.34** (0.07)	75.41 (0.17)		75.31 (17)	75.31 (13)	33.6	**20.68**
6	gr24	1249.82 (0.0)	1249.82 (0.0)		1249.82 (20)	1249.82 (20)	29.31	**22.02**
7	fri26	**635.58** (0.0)	639.87 (7.44)	-	635.58 (20)	635.58 (14)	32.82	**24.51**
8	bays29	**9074.15** (0.0)	9125.15 (107.99)		9074.15 (20)	9074.15 (16)	43.27	**29.09**
9	hk48	**11202.87** (160.19)	11629.01 (260.53)	- -	**11104.67** (5)	11130.68 (1)	123.13	**58.1**
10	eil51	444.56 (6.37)	**441.76** (5.94)		429.51 (1)	**428.98** (1)	157.98	**67.75**
11	berlin52	**7804.2** (172.7)	7879.6 (200.93)		7544.37 (2)	7544.37 (2)	151.52	**65.58**
12	st70	**709.7** (16.1)	716.11 (18.74)		687.17 (2)	**682.57** (1)	245.17	**98.64**
13	eil76	582.44 (9.92)	**572.19** (7.65)	+ +	564.07 (1)	**560.44** (1)	286.36	**120.88**
14	gr96	560.27 (15.81)	**544.71** (14.63)	+ +	531.84 (1)	**521.25** (1)	410.03	**162.77**
15	rat99	1381.41 (42.65)	**1330.27** (33.66)	+ +	1261.47 (1)	**1254.54** (1)	429.69	**168.46**
16	kroa100	23463.91 (967.02)	**22484.26** (641.24)	+ +	21644.63 (1)	**21438.19** (1)	456.37	**176.59**
17	kroB100	24377.23 (825.43)	**23374.96** (398.79)	+ +	22934.09 (1)	**22648.29** (1)	443.89	**189.64**
18	kroC100	22829.39 (966.47)	**22247.78** (580.99)	+	21370.58 (1)	**21120.65** (1)	440.68	**182.34**
19	kroD100	23415.68 (770.93)	**22800.59** (546.97)	+ +	**21847.23** (1)	21927.6 (1)	445.34	**191.01**
20	kroE100	24348.24 (853.96)	**23458.78** (404.8)	+ +	22939.22 (1)	**22600.28** (1)	453.11	**180.97**
21	rd100	8762.08 (240.78)	**8453.56** (216.7)	+ +	8167.28 (1)	**7944.32** (1)	452.18	**174.23**
22	eil101	697.8 (14.06)	**681.02** (10.28)	+ +	675.27 (1)	**663.28** (1)	470.81	**184.65**
23	lin105	16381.44 (620.73)	**15961.57** (431.69)	+	**15112.43** (1)	15213.56 (1)	469.93	**189.41**
24	pr124	65852.22 (2185.46)	**64605.79** (1158.19)	+	**62545.29** (1)	62856.17 (1)	613.59	**258.87**
25	bier127	127725.83 (2163.69)	**124662.81** (3343.96)	+ +	123046.3 (1)	**120291.87** (1)	693	**286.16**
26	ch130	6711.41 (183.02)	**6549.88** (154.37)	+ +	6402.41 (1)	**6309.26** (1)	717.15	**274.32**
27	gr137	810.45 (17.91)	**777.4** (17.43)	+ +	777.4 (1)	**740.8** (1)	748.95	**282.67**
28	pr144	65894.2 (3143.31)	**62675.47** (2276.87)	+ +	60024.85 (1)	**59969.91** (1)	839.24	**325.55**
29	ch150	7302.21 (196.72)	**7179.44** (150.08)	+	6898.65 (1)	**6873.38** (1)	882.93	**363.22**
30	kroA150	29531.52 (726.99)	**28298.44** (457.1)	+ +	28054.71 (1)	**27505.26** (1)	903.16	**359.52**
31	kroB150	29119.96 (682.16)	**28037.59** (501.19)	+ +	27733.8 (1)	**27408.19** (1)	914.42	**381.23**
32	pr152	80544.89 (2146.41)	**76249.77** (1381.28)	+ +	77950.22 (1)	**74166.1** (1)	832.96	**321.93**
33	u159	48256.85 (1576.09)	**45691.99** (1241.51)	+ +	45135.09 (1)	**43827.25** (1)	956.7	**396.28**
34	rat195	2669.82 (48.23)	**2587.59** (50.21)	+ +	2566.11 (1)	**2475.17** (1)	1407.43	**586.83**
35	d198	17222.85 (229.54)	**16577.31** (236.6)	+ +	16616.54 (1)	**16168.11** (1)	1404.75	**573.01**
36	kroA200	32987.91 (898.74)	**32209.01** (550.24)	+ +	31318.23 (1)	**31032.43** (1)	1463.24	**612.19**
38	kroB200	33141.56 (703.43)	**32024.59** (470.22)	+ +	31589.7 (1)	**31175.18** (1)	1481.23	**614.32**
39	gr202	525.04 (8.85)	**513.12** (9.1)	+	508.51 (1)	**499.91** (1)	1501.18	**610.85**
40	ts225	142008.18 (3646.96)	**139670.69** (3189.63)	+	136529.72 (1)	**135246.68** (1)	1838.72	**784.16**
41	tsp225	4345.66 (108.43)	**4280.7** (91.85)	+ +	4181.36 (1)	**4149.42** (1)	1759.1	**753.29**
42	pr226	92600.97 (4465.36)	**88466.55** (2715.16)	+ +	85558.5 (1)	**83753.97** (1)	1794.44	**732.09**
43	gr229	1793.68 (27.74)	**1751.61** (16.54)	+ +	1749.99 (1)	**1721.48** (1)	1897.12	**792.49**
44	gil262	2716.61 (70.9)	**2634** (46.41)	+ +	2573.83 (1)	**2573.19** (1)	2424.19	**968.51**
44	pr264	56196.38 (1166.34)	**54781.23** (1306.87)	+	53795.16 (1)	**52763.29** (1)	2376.31	**984.95**
45	lin318	47548.35 (939.12)	**46843.62** (906.0)	+ +	45882.51 (1)	**44869.69** (1)	3431.37	**1424.16**
	Average	**23969.16**	**23241.30**		**22764.38**	**22404.45**	**812.00**	**334.61**
	Best	**9**	**34**		**4**	**32**	**0**	**45**

particle (at each iteration) makes ESTPSO computationally much expensive in general. But VTPSO applied moderate ST operation on selected particles. Velocity tentative based PS along with ST operation on selected particles makes VTPSO faster convergence and therefore returns good result with less time. As an example, VTPSO took 174.23 seconds for rd100 problem. For the same problem, ESTPSO took more than double time of VTPSO i.e., 452.18 seconds. But VTPSO achieved better result than ESTPSO for the problem; average tour costs achieved by VTPSO and ESTPSO are 8453.56 and 8762.08, respectively. In general, VTPSO took time less than half of ESTPSO to solve a benchmark TSP.

It is also observable from Table I that VTPSO outperforms ESTPSO on the basis of best outcome (i.e., minimum tour cost) from among 20 independent runs. For few small sized problems (e.g., burma14, gr24), both ESTPSO and VTPSO achieved same tour cost in all 20 independent runs. On the other hand, most of the cases, especially for large sized problems, VTPSO achieved better (i.e., lower) minimum tour costs than ESTPSO. Out of 45 cases, VTPSO outperformed ESTPSO for 32 cases; ESTPSO was better than VTPSO for only four cases; and both the methods showed same minimum tour cost for rest nine cases. Among the existing methods, ESTPSO is the best performed PSO based method so far; therefore outperformance of VTPSO over ESTPSO indicates the essence of PS based PSO operation to solve TSPs.

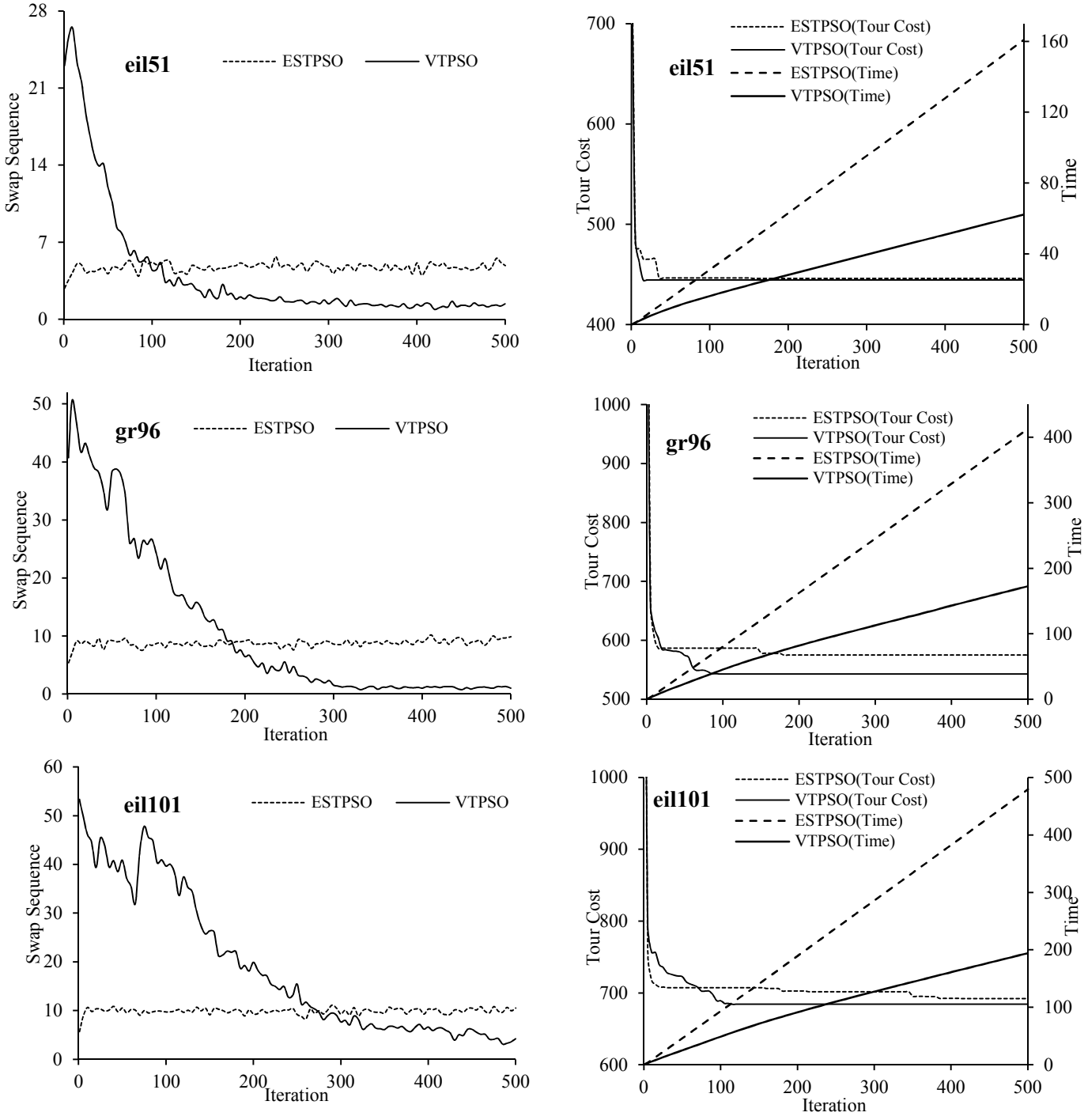
C. Experimental Analyses Comparing with ESTPSO

This section first investigates why VTPSO requires less time than ESTPSO to get the solution on the basis of SS size over iteration. It then investigates the effect of population size (i.e., number particles) and the number of iteration on the performance of ESTPSO and VTPSO. Three problems with different sizes were selected for the analyses and those are eil51, gr99 and eil101.

1) Velocity Swap Sequence and Time over Iteration

Figure 1 presents velocity SS size, global best (i.e., G) solution tour cost and time (in seconds) elapsed from the beginning over iteration for three sample cases of the selected problems. A SS holds several SOs and therefore its size at a particular iteration point is the average number of SOs for all the particles' SS. Since operation of a velocity SS is the collective operations of its individual SOs, a large SS (having many SOs) requires more time than a small one to calculate as well as implement for getting a new tour. It is notable that a particle's solution or tour (X_i) closer to its previous best (i.e., P_i) and/or G generates smaller velocity SS.

ESTPSO applies ST operation out of PSO operations on each particle at each iteration owing to improve each one. Therefore, average velocity SS is maintained at a level throughout iteration in ESTPSO due to such self-improvement besides PSO operations. On the other hand, VTPSO applies ST operation on selected particle with PSO operations and considers PS. Simultaneous operations of both



(a) Swap Sequence vs. Iteration.

(b) Global best Tour Cost and Time vs. Iteration.

Fig. 1. Velocity swap sequence size, global best (G) tour cost and require time (in seconds) elapsed over iteration.

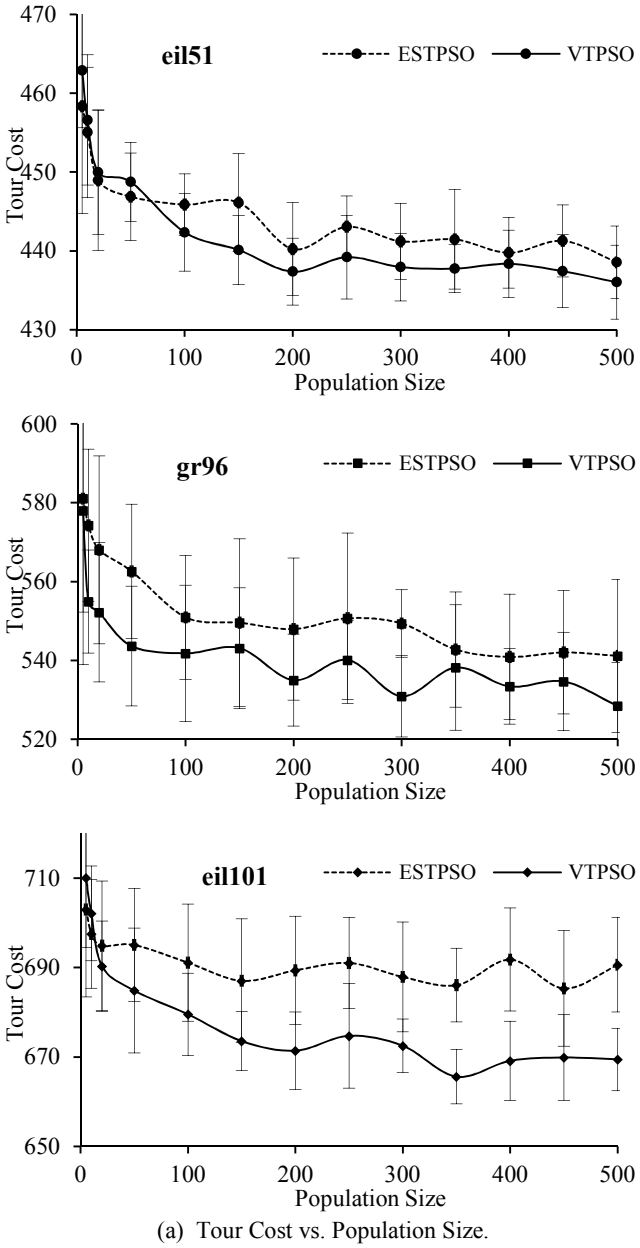
the things might be the reason for velocity SS size reduction (Fig. 1(a)) and faster convergence (Fig. 1(b)) over iterations in VTPSO. At the beginning of iteration, VTPSO seems slower than ESTPSO since SS size of VTPSO is much larger than ESTPSO. But in general, VTPSO is much more time efficient than ESTPSO because of ST operation on selected particles and smaller SS at later iteration period. On the basis of achieved solution, both ESTPSO and VTPSO showed bad result at the beginning and improved over iteration. However, VTPSO outperformed ESTPSO showing better result (i.e., smaller tour cost of *G*) for all the three cases. Finally, partial search and selected ST operations make VTPSO efficient in terms of better TSP outcome as well as required time to give solution.

2) Effects of Population Size and Total Iteration

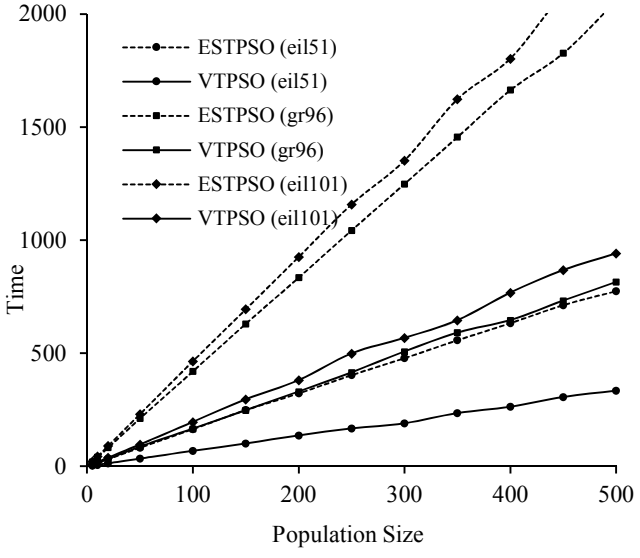
This section investigates the performance of ESTPSO and VTPSO varying population size (i.e., number of particles) and total number of iteration. The results presented in Table I are for the fixed number of population size (=100) and iteration (=500) for all the problems. It is interesting to observe how the algorithms perform on the variation of both the parameters.

Figure 2 shows the achieved tour cost and required time (in seconds) for different population sizes varied from 5 to 500 while total iteration was fixed at 500. The presented results are the average for 10 independent runs. In Fig. 2(a), Standard Deviation (SD) values for 10 runs are placed as vertical bars on the average tour cost. It is seen from the figure that both the methods showed worst tour cost at very small population (e.g., 5) and improved with population size. As an example, for gr96 problem at population size 20, ESTPSO and VTPSO achieved tour costs of 568.04 and 552.18, respectively. For the same problem ESTPSO and VTPSO achieved tour costs of 540.87 and 533.38, respectively, at population size 400. It is common to get better result with larger population but computational time increases much when population size increases as seen in Fig. 2(b). To solve same gr96 problem, ESTPSO took 83.89 and 1665.04 seconds for population sizes 20 and 400, respectively. On the other hand, although VTPSO took more time for larger population but the time it took much less than ESTPSO and found more efficient for larger population. VTPSO took only 33.02 and 646.79 seconds for population sizes 20 and 400, respectively, to solve gr96. At a glance, VTPSO is better than ESTPSO taking less time regardless of population size.

Figure 3 shows the achieved tour cost and required time for different fixed number of iterations varied from 10 to 1000 while population size was fixed at 100. The presented results are the average for 10 independent runs. From Fig. 3 (a) it is observed that both ESTPSO and VTPSO showed the worst tour costs at iteration 10 and improved rapidly up to a certain value (e.g., 200 for eil51) and after that improvement was not significant. As an example, for eil101 problem, ESTPSO and VTPSO achieved tour costs of 709.49 and 722.5, respectively, at iteration 20. For the same eil101 problem, ESTPSO and VTPSO achieved tour costs of 695.14 and 674.16, respectively, at iteration 400. It is notable from the Fig. 3(a) that VTPSO outperformed ESTPSO showing better tour cost but it took less time at any value of iteration as seen in Fig. 3(b). For the 400 iteration of eil101 problem, ESTPSO

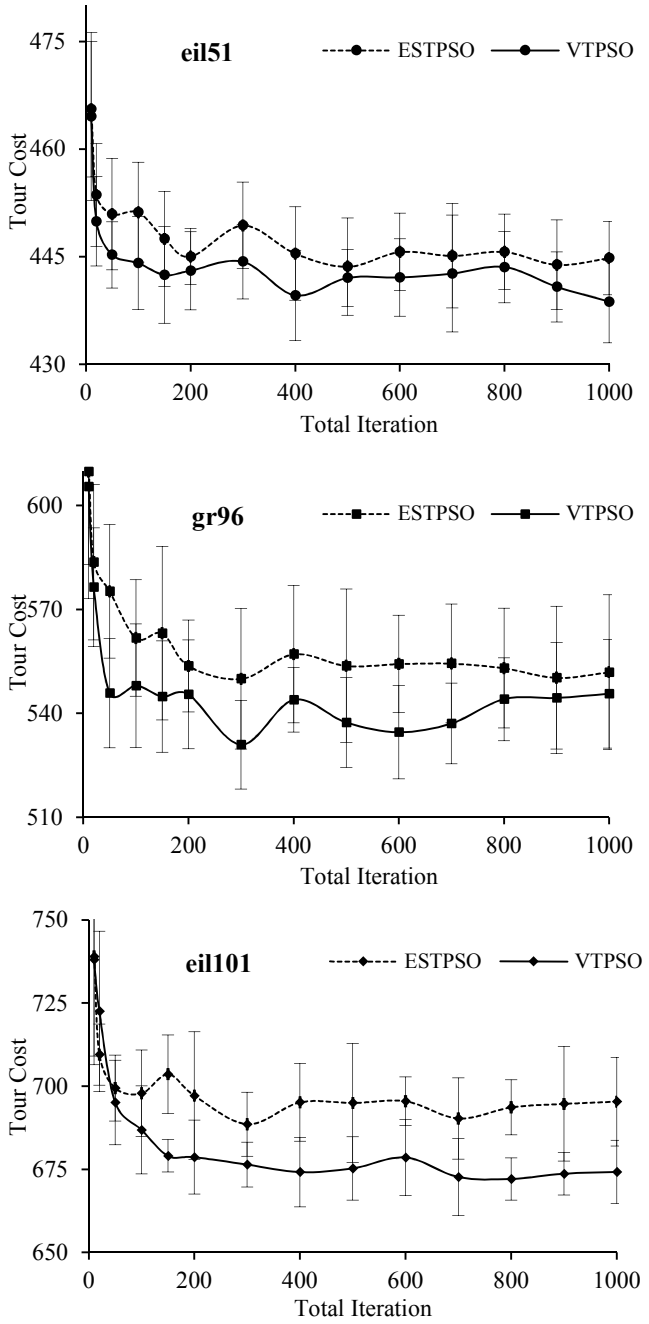


(a) Tour Cost vs. Population Size.

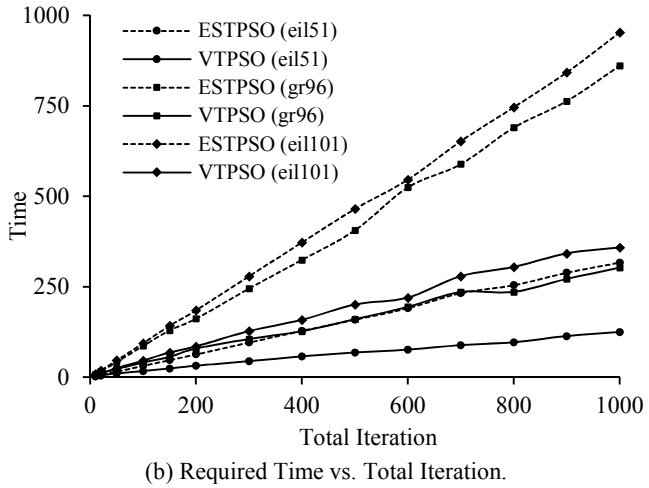


(b) Required Time vs. Population Size.

Fig. 2. Variation effect of population size on tour cost and require time.



(a) Tour Cost vs. Total Iteration.



(b) Required Time vs. Total Iteration.

Fig. 3. Variation effect of fixed total iteration on tour cost and require time.

took 372.78 seconds; but VTPSO took 158.57 seconds that is less than half of ESTPSO. Finally, the Figs. 2 and 3 clarified VTPSO as a good method to solve TSP.

D. Experimental Results Comparing with ACO and PSM

While VTPSO is shown to outperform the prominent PSO based method ESTPSO, this section compares performance of VTPSO with other prominent methods for TSP for better understanding. We have considered Ant Colony Optimization (ACO) [22-23] and Producer-Scrounger Method (PSM) [24] to compare the outcome of proposed VTPSO with the methods. ACO is the pioneer as well as most popular swarm intelligence based method for TSP. On the other hand, PSM is the most recent method to solve TSP. ACO, PSM and VTPSO have been tested on a suite of 20 benchmark problems with fair settings. For the fair comparison, the number of iteration was set at 500 for the algorithms. The number of ants in ACO was equal to the number of cities as it desired. On the other hand, the population size was varied from 100 to 200 for PSM and VTPSO. In ACO, *alpha* and *beta* were set to 1 and 3, respectively. On the other hand, the *RNC* (rate of near cities consideration) for producer scanning in PSM was set to 0.1. The selected parameters were not optimal values, but considered for simplicity as well as for fairness in comparison. The experiments performed on the same computer described earlier.

Table II compares performance of ACO, PSM and VTPSO for solving the benchmark TSPs on the basis of 20 independent runs to solve a problem with a method. For a particular problem, the best tour cost (i.e., smallest value) among the three algorithms is shown in bold-face type and worst one (i.e., largest value) is shown in underlined face type. Bottom of the table shows the achieved average tour cost and best/worst summary (which indicates on how many problem instances a method gave best/worst result) for all 20 problems by the methods. A Win/Draw/Loss summary of the results is also presented for better understanding. Pair two tailed *t*-test was conducted to determine the significance in the variation of results of VTPSO with ACO and PSM. If tour cost of VTPSO was found significantly better than ACO/PSM by *t*-test for a particular problem, it is marked with a plus (+) sign in the column of *t*-test evaluation. On the other hand, a minus (−) sign indicates VTPSO was significantly worse than ACO/PSM for a particular problem. A single plus/minus means the tour cost difference was statistically significant with 95% confidence interval and a double plus/minus is for 99% confidence interval.

The average tour costs presented in the Table II indicate that VTSPSO is the best and PSM is the worst. The average tour cost over all 20 problems was 12745.46 for VTPSO, the value is the best among the three methods. The achieved average tour costs for ACO and PSM were 14026.29 and 15147.38, respectively. PSM is shown worst tour costs for 15 problems out of 20 cases showing best for none. In pair Win/Draw/Loss comparison, PSM is better than ACO for five cases only. ACO is found best for only one problem (i.e., gr17) and worst for six cases. On the other hand, proposed VTPSO is shown best for 19 cases but worst for none. In pair Win/Draw/Loss comparison, VTPSO is better than ACO and PSM for 19 and all 20 cases, respectively. Moreover, *t*-test

TABLE II
COMPARISON OF THE EXPERIMENTAL RESULTS OF THE PROPOSED VTPSO WITH ACO AND PSM TO SOLVE BENCHMARK TSPS.

Sl.	Problem	Average Tour Cost (Standard Deviation)			*t*-test eval. of VTPSO		Minimum Tour Cost		
		ACO	PSM	VTPSO	ACO	PSM	ACO	PSM	VTPSO
1	burma14	31.31 (0.24)	30.9 (0.1)	**30.87** (0.0)	++		31.21	**30.87**	**30.87**
2	ulysses16	77.13 (0.0)	74.22 (0.31)	**74.0** (0.02)	++	++	77.13	**73.99**	**73.99**
3	gr17	**2332.58** (0.0)	2350.4 (44.76)	2337.2 (13.86)			2332.58	2332.58	2332.58
4	gr21	2955.42 (0.0)	2966.43 (250.07)	**2681.41** (27.42)	++	++	2955.42	**2672.27**	**2672.27**
5	ulysses22	86.19 (0.07)	78.09 (3.7)	**75.41** (0.17)	++	+	86.08	**75.31**	**75.31**
6	gr24	1267.13 (0.0)	1367.3 (97.51)	**1249.82** (0.0)	++	++	1267.13	1251.33	**1249.82**
7	fri26	646.39 (0.37)	697.64 (45.95)	**639.87** (7.44)	++	++	644.8	**635.58**	**635.58**
8	bays29	9964.78 (0.0)	9681.97 (377.14)	**9125.15** (107.99)	++	++	9964.78	**9074.15**	**9074.15**
9	hk48	12723.18 (68.36)	13188.05 (805.78)	**11483.65** (256.08)	++	++	12699.86	11757.54	**11104.67**
10	eil51	502.26 (9.96)	486.34 (29.97)	**442.51** (5.98)	++	++	461.42	445.81	**428.86**
11	berlin52	8061.57 (48.94)	8800.91 (525.06)	**7863.06** (214.99)	++	++	7870.45	7806.24	**7544.37**
12	st70	745.11 (7.86)	850.77 (53.89)	**716.11** (18.74)	++	++	734.19	743.75	**682.57**
13	eil76	598.87 (7.23)	629.12 (25.78)	**566.25** (7.42)	++	++	583.28	586.67	**554.64**
14	gr96	588.76 (9.15)	607.7 (31.26)	**537.38** (15.03)	++	++	564.37	556.64	**515.12**
15	rat99	1368.95 (1.08)	1471.25 (66.2)	**1330.27** (33.66)	++	++	1366.3	1366.58	**1254.54**
16	kroa100	24662.12 (81.77)	29134.2 (2087.6)	**22388.06** (645.9)	++	++	24524.53	24872.86	**21399.53**
17	kroB100	25369.53 (508.8)	30759.27 (2275.7)	**23211.26** (499.27)	++	++	24675.03	26641.31	**22305.35**
18	kroC100	23293.17 (105.0)	28882.39 (2260.0)	**22278.36** (659.16)	++	++	23248.13	24741.26	**21063.05**
19	rd100	9420.6 (58.82)	10389.55 (749.11)	**8453.56** (216.7)	++	++	9210.67	9134.8	**7944.32**
20	eil101	738.64 (4.91)	764.67 (33.91)	**672.32** (9.33)	++	++	729.95	722.88	**657.62**
	Average	**14026.29**	**15147.38**	**12745.46**			**13965.82**	**13432.35**	**12293.84**
	Best/Worst	**1/6**	**0/15**	**19/0**			**1/13**	**6/7**	**19/0**

Method	Pairwise Win/Draw/Loss Summary on Average Tour Cost and Minimum Tour Cost						
	ACO	PSM	VTPSO		ACO	PSM	VTPSO
ACO	-	5/0/15	19/0/1		-	13/1/6	19/1/0
PSM		-	20/0/0			-	13/7/0

shows that the performance of VTPSO was significantly better than ACO and PSM on 19 and 18 problems, respectively.

On the basis of minimum tour costs presented in the Table II, VTPSO is the best and ACO is the worst. ACO is the prominent method for solving TSP and considers population size as the number cities of a given problem. ACO starts placing different ants in different cities and its initialization does not differ among individual runs [21-22]. Therefore, the tour costs achieved by ACO in different runs are found consistent showing lower SD values. For several problems, especially small sized ones (e.g., ulysses16, gr17, gr21), ACO was shown same tour cost in all 20 individual runs and therefore SD of average tour cost is shown as zero for the problems in the table. For any problem, PSM gave most variant outcomes among different runs showing largest SD value. Consequently, PSM was shown to outperform ACO on the basis of minimum tour cost from 20 runs. PSM achieved minimum tour cost better than ACO in 13 cases, in which six cases PSM was the best along with proposed VTPSO. However, PSM was shown worst for seven cases on the basis minimum tour cost. On the other hand, VTPSO was shown to achieve best minimum tour cost for all the cases except gr17; gr17 is a small sized problem and all three methods achieved equal minimum tour cost of 2332.58 for it. Moreover, according to Win/Draw/Loss summary, VTPSO is better than ACO and PSM for 19 and 13 cases, respectively; rest of the cases VTPSO is shown same tour cost of ACO and PSM, i.e., one and seven cases, respectively. Finally, proposed VTPSO seems significantly better than ACO and PSM to solve TSP.

E. Effects of Population Size and Iteration Comparing with ACO and PSM

This section investigates the performance of ACO, PSM and VTPSO varying population size (i.e., number of individuals) and number of iteration. The experiments performed on the same machine explained before. Three problems with different sizes (i.e., eil51, gr96 and eil101) were selected for the analyses.

Figure 4 shows the achieved tour cost for different population sizes varied from 5 to 500 for PSM and VTPSO. Number of ants in ACO was equal to the number of cities of a particular problem as it desired. The presented results are the average for 10 independent runs for fixed 500 iteration as termination criteria; SD values for the runs are placed as vertical bars on the average tour cost. Since population size was fixed in ACO for a particular problem it showed invariant performance in the figure. While ACO is unable to work with different population size, PSM and VTPSO may outperform ACO varying population size. For eil51 problem, ACO was shown tour cost of 504.03; but both PSM and VTPSO outperformed ACO at any population size. PSM and VTPSO were shown tour costs of 479.37 and 442.35, respectively, at population size 100. However, PSM was also found less invariant with population size as seen in Fig. 4. Because PSM is producer (the single best solution) centric; and it is reported that PSM may works well and gives suitable result with relatively small population size [23]. On the other hand, VTPSO seems to perform well with population variation and may outperform ACO and PSM. For gr96 problem, as an example, VTPSO achieved tour costs of 577.97 and 534.96

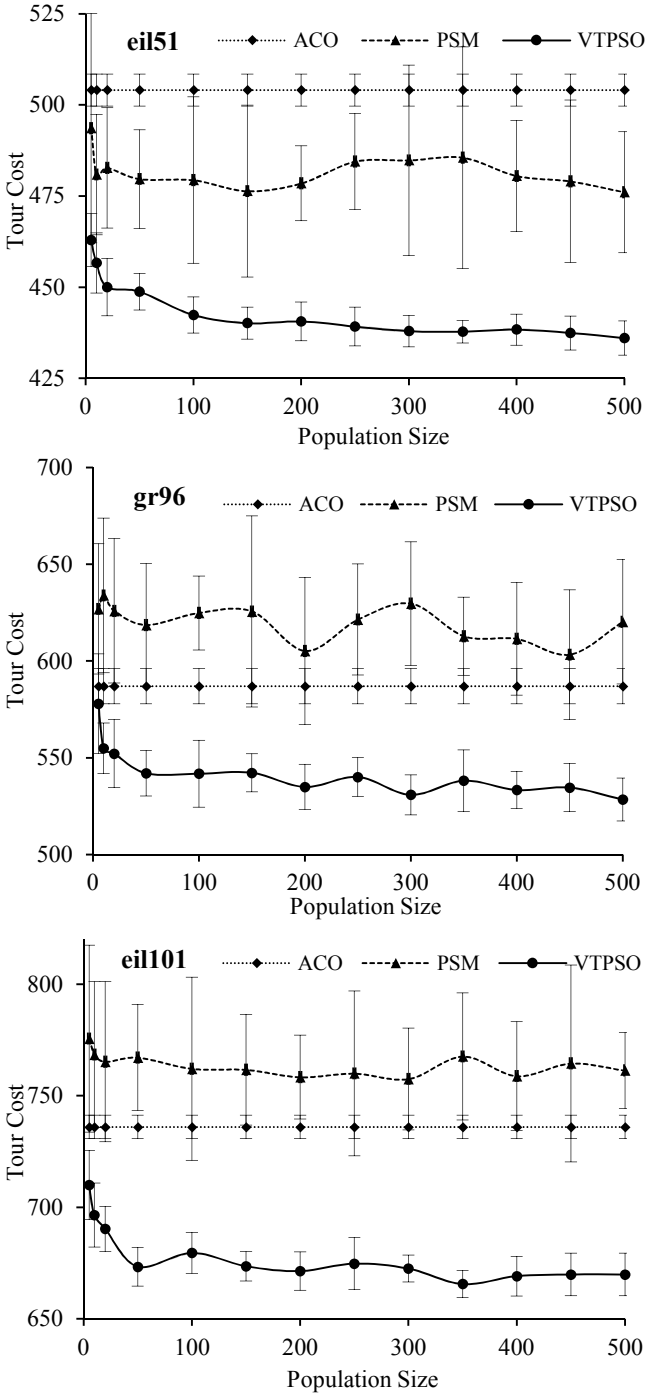


Fig. 4. Variation effect of population size on tour costs of ACO, PSM and VTPSO.

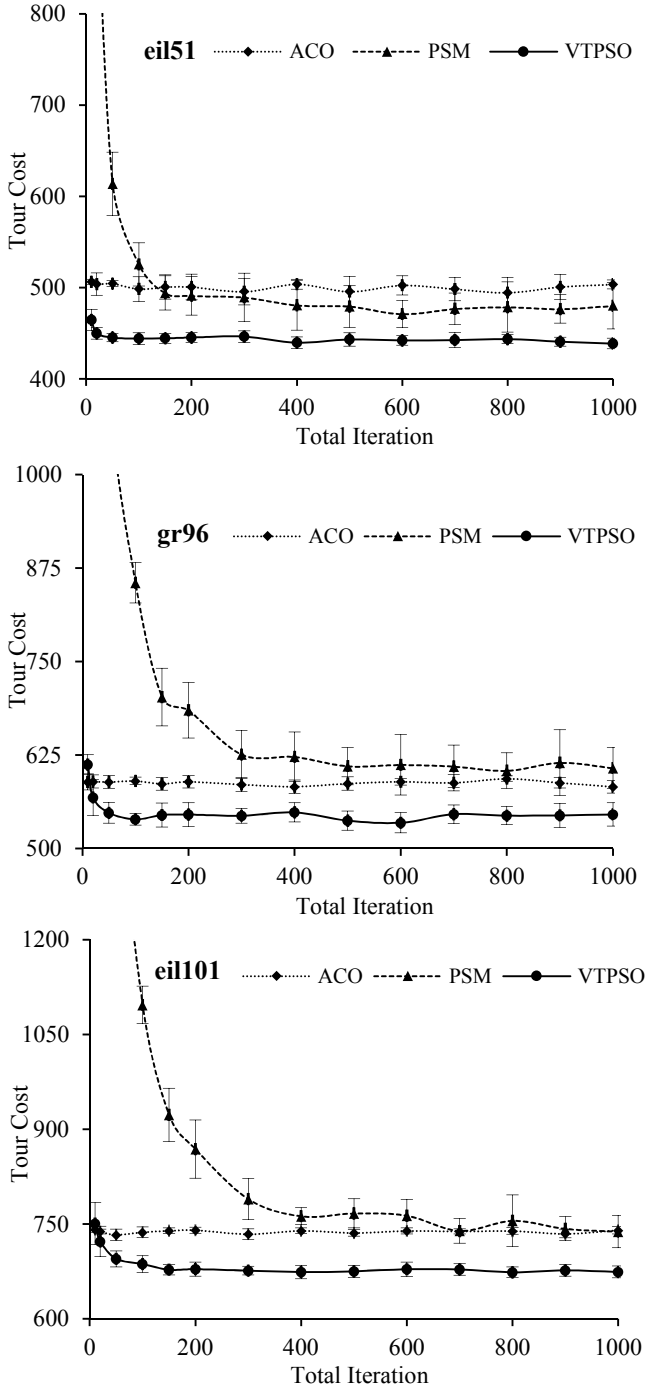


Fig. 5. Variation effect of total number of iteration on tour costs of ACO, PSM and VTPSO.

for population sizes of 5 and 200, respectively. For the same problem, ACO achieved tour cost of 586.98 and the best tour cost of PSM was 603.32 at population size 450. At a glance, proposed VTPSO has shown the ability to achieve better result varying population size.

Figure 5 compares performance among ACO, PSM and VTPSO for iteration variation; iteration varied from 10 to 1000. In the experiments, population size was fixed at 100 for PSM and VTPSO; and the number ants in ACO was equal to the number of cities of a particular problem. The presented results in figures are the average for 10 independent runs; SD values are also placed as vertical bars on the average tour cost. According to the results presented for the problems in the figure, all the methods are shown to perform worse for small number of iteration. However, ACO is shown the most invariant and PSM is shown the most variant in performance for iteration variation. As an example, for eil101 problem at iteration 10, the achieved tour costs were 741.37, 2458.78 and 750.81 for ACO, PSM and VTPSO, respectively. For the same problem, the best tour costs were 733.15 for ACO (at 50 iteration) and 738.17 for PSM (at 1000 iteration). On the other hand, VTPSO was shown the best tour cost of 674.16 (at 400 iteration) and the achieved tour cost is much better (i.e., lower) than the achieved best values of ACO and PSM. With larger variation for iteration variation, PSM was inferior to ACO for small number iterations while it was outperformed ACO for larger iteration. As an example, for eil51 problem, ACO was better than PSM until iteration 100 but after that PSM outperformed ACO. However, proposed VTPSO always outperformed ACO and PSM for any value

of iteration for eil51. The similar performance of the proposed VTPSO for gr96 and eil101 problems reveals that VTPSO is an effective method for TSP.

V. Conclusions

TSP is a popular combinatorial optimization problem and interest grows in recent years to solve it new ways. This study investigated a new PSO based method, called Velocity Tentative PSO (VTPSO), to solve TSP. In VTPSO, each particle represents a complete tour and velocity is measured as a Swap Sequence (SS) consisting of several Swap Operators (SOs). In the conventional existing methods, a new tour is considered after applying a complete velocity SS with all its SOs. In contrast, proposed VTPSO considered calculated velocity SS as the tentative velocity; checked all the tentative solutions applying SOs of the SS one after another sequentially; and picked the best solution among the tentative solutions. VTPSO has been tested on a large number of benchmark TSPs and it outperformed ESTPSO, the prominent PSO based method. More interestingly, VTPSO took less than half time of ESTPSO to solve a benchmark TSP, in general. The reason behind the less time requirement by VTPSO has been revealed from the experimental analyses; VTPSO applied self-tentative operation on selected particles and its velocity SS size reduced over iteration. Moreover, VTPSO also compared with two other prominent methods (i.e., ACO and PSM) for TSP and outperformed both of the methods in solving benchmark TSPs.

A potential future direction is also opened from this study. This study considered partial search maintaining the sequence of SOs in the velocity SS and identified that a portion of SS may give better outcome than whole SS implementation. It is notable that SOs may be applied independently without sequence because velocity SS of a particle comes from three different sources: its previous velocity, difference with previous best solution and difference with global best solution. Algorithm development with such consideration may give better result and remain as future study.

References

[1] R. Eberhart and J. Kennedy, "A New Optimizer Using Particles Swarm Theory," in *Proc. Sixth International Symposium on Micro Machine and Human Science,* Nagoya, Japan, October 1995, pp. 39–43.

[2] 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, no. 3, pp. 281–295, 2006.

[3] 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,* vol. 7, no. 1, pp. 109–124, 2007.

[4] L. Chuang, Y. Lin, and C. Yang, "Data Clustering Using Chaotic Particle Swarm Optimization," *IAENG International Journal of Computer Science*, vol. 39, no. 2, pp. 208–213, 2012.

[5] Z. Zhong and D. Pi, "Forecasting Satellite Attitude Volatility Using Support Vector Regression with Particle Swarm," *IAENG International Journal of Computer Science*, vol. 41, no. 3, pp. 153–162, 2014.

[6] Hong Zhang, "An Analysis of Multiple Particle Swarm Optimizers with Inertia Weight for Multi-objective Optimization," *IAENG International Journal of Computer Science*, vol. 39, no. 2, pp. 190–199, 2012.

[7] Y. F. Liao, D. H. Yau and C. L. Chen, "Evolutionary algorithm to traveling salesman problems," *Computers & Mathematics with Applications, Elsevier Publisher,* vol. 64, no. 5, pp. 788–797, 2012.

[8] 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 2008.

[9] 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, no. 6-2, pp. 264–271, 2012.

[10] 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*, November 2003, pp. 1583–1585.

[11] 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.

[12] 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.

[13] 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," *Information Processing Letters*, vol. 103, pp. 169–176, 2007.

[14] H. Fan, "Discrete Particle Swarm Optimization for TSP based on Neighborhood," *Journal of Computational Information Systems (JCIS),* vol. 6, pp. 3407–3414, 2010.

[15] W. Zhong, J. Zhang and W. Chen, "A Novel Discrete Particle Swarm Optimization to solve Traveling Salesman problem," *IEEE Congress on Evolutionary Computation*, 2007, pp. 3286–3287.

[16] M. F. Tasgetiren, P. N. Suganthan and Q. Pan, "A Discrete Particle Swarm Optimization Algorithm for the Generalized Traveling Salesman Problem," in *Proc. 9th annual conference on Genetic And Evolutionary Computation*, 2007, pp. 158–167.

[17] E. F. G. Goldbarg, M. C. Goldbarg and G. R.de Souza, "Particle Swarm Optimization Algorithm for Traveling Salesman Problem," *Traveling Salesman Problem, Federico Greco(Ed.)*, InTech, 2008.

[18] 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, June 14–15, 2011, pp. 24-1–24-8.

[19] R. Matai, S. Singh and M. L. Mittal, "Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches," *Edited by D. Davendra, InTech,* 2010, pp. 1–24.

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

[21] E. Bonabeau, M. Dorigo and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, Oxford, 1999.

[22] O. Cordon, F. Herrera, T. Stutzle, "A review on the ant colony optimization metaheuristic: basis, models and new trends," *Mathware and Soft Computing*, vol. 9, pp. 141–175, 2002.

[23] 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 (IJISA)*, vol. 7, no. 3, pp. 29–36, 2015.