A Comparative Study of Prominent Particle Swarm Optimization Based Methods to Solve Traveling Salesman Problem

Akhand MAH*, Shaik Imran Hossain and Shahina Akter

Department of Computer Science and Engineering, Khulna University of Engineering and Technology Khulna, Bangladesh

Abstract

Computational methods inspired by natural phenomenon have gain much interest in the recent years. Among the developed algorithms, particle swarm optimization (PSO), mimicking behavior of bird flocking or fish schooling, seems the most famous method due to its simplicity as well as performance. A variant number of PSO based methods was developed for traveling salesman problem (TSP), the most popular combinatorial problem. The aim of the study is to make a comparative study of several prominent PSO based methods in solving TSP. The study is important because different PSO based methods have been developed by different researchers and tested on different sets of problems. Therefore, the description of the prominent PSO based methods in a similar fashion reveals distinct features of individuals. Moreover, experimental results on a common benchmark TSP data set will reveal performance of each method. In this study, the methods have been tested on a large number of benchmark TSPs and outcomes compared among themselves as well as ant colony optimization (ACO), the prominent method to solve TSP. Experimental results revealed that enhanced self-tentative PSO (ESTPSO) and velocity tentative PSO (VTPSO) outperformed ACO; and self-tentative PSO (STPSO) is competitive to ACO. On the other hand, experimental analysis revealed that ESTPSO is computationally heavier than others and VTPSO took least time to solve a benchmark problem. The reasons behind performance and time requirement of each individual method are explained and VTPSO is found most effective PSO based method to solve TSP.

Keywords: Particle swarm optimization; Traveling salesman problem; Swarm intelligence

Introduction

Computational methods inspired by natural phenomenon have gain much interest in the recent years. The natural phenomenon from microscopic matter to living behavior of insects or large animals is studied to develop distinct computing techniques in last few decades. Genetic algorithm is the pioneer method inspired by biological evolution and successfully applied in solving numerous tasks [1-4]. In the last decade, collective behaviors of various insects (e.g. ant, bee, fish, firefly, glow worm) have been investigated and a number of computing methods in the category of swarm intelligence (SI) have been proposed [5-14]. Recently, living and hunting behaviors of animals (e.g. cattle, gray wolf, lion) are considered to develop better SI based optimization techniques [15,16]. Among the developed algorithms, particle swarm optimization (PSO) [10] seems the most popular now days due to its simplicity as well as performance which is based on behavior of bird flocking or fish schooling.

PSO was developed for function optimization as like most of the optimization methods and it was shown better or at least competitive to any other methods in solving various optimization tasks [10,17-19]. Later on, PSO has been transformed to tackle different combinatorial optimization tasks including traveling salesman problem (TSP) [20-27]. Among various combinatorial problems, TSP is the most important problem having many real world applications [28]. Considering TSP as a general test bench, a number of developed methods have been applied to solve TSPs to identify their performance; among them ant colony optimization (ACO) is the prominent one [8,29]. A number of PSO based methods with different modifications have also been developed to solve TSP [11,20-26].

The aim of the study is to make a comparative study of several popular PSO based methods to solve TSP. The study is important because different PSO based methods have been developed by different researchers and tested on different sets of problems. Therefore, the...
considering its previous best position and the best one among all the particles in the population. The processes continue until the stopping criterion is reached. PSO has been successfully applied on numerous continuous and combinatorial optimization tasks [10,17-27].

PSO was developed for continuous problems (e.g. function optimization) in which particles moves in multi-dimensional search space to reach optimal position [10]. If a particle resides in $X_i = [x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iD}]$, $P_i = [p_{i1}, p_{i2}, \ldots, p_{iD}]$ is its previous best position and $G = (g_1, g_2, \ldots, g_D)$ is the global best position of entire population, the particle calculates its velocity $V_{i1}, V_{i2}, \ldots, V_{iD}$ according to the following equation.

$$V_i^{(t)} = \omega V_i^{(t-1)} + c_1 \cdot r_1 \cdot \left( P_i^{(t)} - X_i^{(t-1)} \right) + c_2 \cdot r_2 \cdot \left( G^{(t)} - X_i^{(t-1)} \right)$$  \hspace{1cm} (1)

In the equation $\omega$ is inertia factor, $c_1$ and $c_2$ are learning factors, $r_1$ and $r_2$ and are vectors of random values between (0,1). After that the particle moves to a new position adding the calculated velocity with its position according to Eq. (2).

$$X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)}$$  \hspace{1cm} (2)

Algorithm 1 shows the steps of PSO algorithm. The number of particles in PSO is a user defined parameter; and each particle is assigned with a random solution and a random velocity in initialization step (Step 1). At this initial stage, $P$ is considered as the assigned random tour. On the other hand, global best solution $G$ is defined as the best one based on the fitness. At every iteration step, for each particle, PSO calculates a new velocity value using Eq. (1) which is used to find the next position of the particle using Eq. (2).

Here, $\omega$ means to apply SO on the solution $S$ [11]. Swap Operator is the most important in solving TSP problem and its operation is similar to mutation operation of genetic algorithm [1,2].

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

$$SS_j = (SO_{i1}, SO_{i2}, SO_{i3}, \ldots, SO_{iN})$$  \hspace{1cm} (3)

Where $(SO_{i1}, SO_{i2}, SO_{i3}, \ldots, SO_{iN})$ are Swap Operators. Implication of a SS on a solution is nothing but implication of all its SOs maintaining order. Eq. (4) shows solution $S_j$ is achieved applying $SS_j$ on $S_i$.

$$S_j = S_i \odot SS_j = S_j + (SO_{i1}, SO_{i2}, SO_{i3}, \ldots, SO_{iN})$$  \hspace{1cm} (4)

The implication order of SOs of $SS_j$ is important because implication of same SOs in different order may give different solutions from the original solution. $SS_j$ may also be calculated from solutions $S_j$ and $S_i$ in the following equation:

$$SS_j = S_j \odot SS_i = (SO_{i1}, SO_{i2}, SO_{i3}, \ldots, SO_{iN})$$  \hspace{1cm} (5)

Furthermore, operator defines merging operation of several SSs to get a new SS [11]. If $SS_1 = SO(1,3), SO(5,4)$ and $SS_2 = SO(5,2), SO(4,3)$ then new Swap Sequence SS(new) merging $SS_1$ and $SS_2$ is

$$SS(new) = SS_1 \odot SS_2$$

It is notable that outcome with different SSs may be same even on applying on a solution. Among these SSs which have the least SSs is called Basic Swap Sequence (SSS). As an example, when SSS, $SS_1 = SO(1,2), SO(2,1), SO(1,3), SO(2,3), SO(4,5)$ and $SS_2 = SO(1,3), SO(5,2), SO(4,3)$ then

$$SS_1 = S_j \odot SS_1$$

individually, the outcome is $S_j = (a-c-e-b-d)$ independently. Therefore, $SS_2$ is the Basic SS. It is also found by using Eq. (5), i.e., $S_j = SS_1$.

Prominent PSO based methods to solve TSP

The work of [11] is the basic SS based PSO (SSPSO) method to solve TSP transforming PSO operations of function optimization to TSP. Introducing additional operations with SSPSO other algorithms to solve TSP are self-tentative PSO (STPSO) and Enhanced Self Tentative PSO (ESTPSO) [21]. STPSO introduces tentative behavior in SSPSO that tries to improve each particle placing a node in a different position. ESTPSO also considered block node adjustment in addition to individual node adjustment of STPSO [24]. Most recently, Velocity Tentative PSO (VTPSO) introduced tentative operation with velocity implementation [26]. The algorithms follow common initialization technique like standard PSO: consider user defined number of particles, assign random tour ($X$) and velocity ($V$) to each individual particle, calculate fitness of each tour, consider $P$ as $X$, and assign $G$ as the best tour among those tours. On the other hand, the tour that belongs to $G$ is commonly considered as an outcome in any algorithm. The algorithms differ among themselves in velocity SS calculation, new tour generation.
and additional operations with PSO operations. The following subsections briefly explain the methods.

SSPSO [11] is the pioneer PSO based method to solve TSP considering SS as a velocity operator to transform a tour to a new one applying all its SOs. The velocity SS of a particle is measured on its previous best tour \( P_i(t-1) \) and the global best tour \( (G^{(o-i)}) \) in the population using Eq. (7).

\[
V_i(t) = V_i(t-1) + \alpha(G^{(t-1)} - X_i(t-1)) + \beta(G^{(t-1)} - X_i^{(t-1)}) + \omega \delta(1,0) \tag{7}
\]

\[
X_i(t) = X_i(t-1) + V_i(t) \tag{8}
\]

\[
P_i(t) \quad \text{and} \quad (G^{(o-i)}) \quad \text{will be considered on velocity calculation selecting more SOs.}
\]

Algorithm 1: Particle Swarm Optimization (PSO)
Step 1: Initialization
Step 2: Calculate the velocity and update position of the particles according to Eq. (1) and Eq. (2).
Step 3: Update \( P_i \) if the new solution \( X_i^o \) is superior to \( P_i \).
Step 4: Update \( G \) if the new solution \( X_i^o \) is superior to \( G \).
Step 5: Loop to Step 2 until a termination criterion is met.
Step 6: Take the global best solution \( G \) as an outcome.

In Eq. (7), \( \alpha, \beta \) are random number between 0 and 1, \( \alpha \) \((P_i(t) - X_i(t))\) means all SOs in BSS for \( (P_i(t) - X_i(t)) \) should be maintained with the probability of \( \alpha \), it is the same for \( (G^{(t)} - X_i^{(t-1)}) \). The bigger the value of \( \alpha \) and \( \beta \) the greater the influence of \( P_i(t) \) and \( (G^{(o-i)}) \) will be considered on velocity calculation selecting more SOs.

Algorithm 2: Swap Sequence based PSO (SSPSO)
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. (8).
c. Update \( P_i \) if the new solution \( X_i^o \) is superior to \( P_i \).
d. Update \( G \) if the new solution \( X_i^o \) 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.

After that each particle moves to a new tour \( (X_i^o) \) applying velocity SS on its previous solution \( (X_i^{(t-1)}) \) using Eq. (8).

The steps of SSPSO are shown in Algorithm 2. As like any other population based algorithm SSPSO initializes (Step 1) user defined number of particles with random solutions (i.e., TSP tour). At this stage, a random velocity SS is assigned to each particle; \( P_i \) is considered as the current random tour and \( G \) is the best tour among them. Step 2 is for the main operation of SSPSO using Eq. (7) and Eq. (8). The method checks termination criterion in each iteration in Step 3. Usually a maximum number of iterations are considered as the termination criteria. If termination criterion meets then \( G \) is considered as the outcome (Step 4); otherwise it loops back to Step 2 for further operation.

STPSO [21] introduces tentative behavior after PSO operation in solving TSP owing to improve each particle. The steps of STPSO algorithm to solve the TSP are shown in Algorithm 3. Step 2 of STPSO is for PSO operations and is similar to SSPSO. The method calculates particle’s velocity as of Eq. (9) and updates position similar to SSPSO using Eq. (8).

Algorithm 3: Self-Tentative PSO (STPSO)
Step 1: Initialization
Step 2: For each particle \( X_i \) in the swarm
a. Calculate velocity \( V_i(t) \) according to Eq. (9).
b. Update solution using Eq. (8).
c. Update \( P_i \) if the new solution \( X_i^o \) is superior to \( P_i \).
d. Update \( G \) if the new solution \( X_i^o \) is superior to \( G \).
Step 3: Tentative Operation on Each Particle \( X_i \)
   a. Single Node Adjustment
   b. Update \( P_i \) if the new solution \( X_i^o \) is superior to \( P_i \).
   c. Update \( G \) if the new solution \( X_i^o \) is superior to \( G \).
   d. Update \( G \) if the new solution \( X_i^o \) 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.

\[
V_i(t) = \omega V_i^{(t-1)} + C_1 r_1 \left( P_i(t-1) - X_i(t-1) \right) + C_2 r_2 \left( G^{(t-1)} - X_i^{(t-1)} \right) + \omega \delta(1,0) \tag{9}
\]

In Eq. (9), \( C_1 \) and \( C_2 \) are learning factors, and \( r_1 \) and \( r_2 \) are vectors of random values between (0,1). The scaling factor \( \omega \) defines the portion of previous velocity in the current velocity.

Self-Tentative operation in STPSO (Step 3) is mainly single node adjustment. 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 [23]. Outcome of this Self-Tentative operation is better particle solution (i.e., TSP tour) if any single node adjustment is able to improve its fitness. However, single node adjustment of STPSO might not be sufficient to get optimal result in some cases [22].

ESTPSO [22] is an extension of STPSO with block node adjustment to get better result overcoming limitation of single node adjustment. The steps of ESTPSO algorithm are shown in Algorithm 4.

Algorithm 4: 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. (9).
b. Update solution using Eq. (8).
c. Update \( P_i \) if the new solution \( X_i^o \) is superior to \( P_i \).
d. Update \( G \) if the new solution \( X_i^o \) is superior to \( G \).
Step 3: Tentative Operation on Each Particle \( X_i \)
   a. Single Node Adjustment
   b. Block Node Adjustment
   Update \( P_i \) if the new solution \( X_i^o \) is superior to \( P_i \).
   Update \( G \) if the new solution \( X_i^o \) 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.

In ESTPSO, the block node adjustment (Step 3b) after single node adjustment (Step 3a) is only the addition to STPSO. In block node adjustment, position of a block of nodes of a particle is altered for better fitness of particle tour. Since selection of block length is hard to determine 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 \( k \) and \( k_{max} \). The value of \( k_{max} \) changes according to the generation and becomes longer with the generation [24]. Detailed description regarding block node adjustment is available in the existing studies [22,23]. Finally, the termination criteria of ESTPSO are similar to SSPSO and STPSO.
VTPSO [26] is the most recent PSO based method conceiving a moderate way (called partial search) of getting new tour with velocity SS while calculates velocity similar to other methods (e.g. SSPSO, STPSO). Moreover, it also employs a moderate self-tentative technique to improve its performance.

Algorithm 5 shows the steps of the VTPSO which initializes the population with random solutions. At each iteration step, VTPSO calculates velocity SS (Step 2a) using Eq. (10) that is simpler than Eq. (9) of STPSO/ESTPSO. Eq. (10) does not have any parameter to set and \( a, \beta \) are random numbers between 0 and 1.

**Algorithm 5:** Velocity Tentative PSO (VTPSO)

**Step 1:** Initialization

**Step 2:** For each particle \( X_i \) in the swarm

Calculate velocity \( V_i(t) \) using Eq. (10)

Update \( X_i(t) \) through Partial Search manner

Apply Tentative Operation on \( X_i(t) \) if it is superior to \( P_i \)

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 \( G \)

**Step 3:** Loop to Step 2 until a termination criterion is met

**Step 4:** Take the global best solution \( G \) as an outcome

\[
V_i(t) = V_i(t-1) + \alpha \left( P_i - X_i(t-1) \right) + \beta \left( G - X_i(t-1) \right)
\]

(10)

The partial search (PS) technique, to apply calculated SS to update particle’s position (i.e., TSP tour) is the main difference of VTPSO from other methods. In PS technique (Step 2b), performance of a tour is measured after applying each of SS of SS 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. It is reported that PS technique may explore better result evaluating intermediate tentative tours without increasing the computational time. VTPSO applies self-tentative operation (Step 2c) on tour \( \{X_i\} \) when it is found better than \( P_i \). The detail description of PS technique and VTPSO is available in [26]. Finally, the termination criteria of VTPSO are similar to other methods.

**Experimental Studies**

This section first gives a short description of the benchmark TSPs and experimental settings. After that experimental result of the PSO based methods are presented and comparison among them is made. Since ACO is the most prominent method for TSP [8,9,29], it is also considered in this study to identify the effectiveness of a PSO based method with respect to it. This section also presents an experimental analysis to identify variation effect of parameter values on performance.

**Benchmark problems and experimental setup**

In this study, a suite of 58 benchmark problems are considered from TSPLIB [38] where number of cities varied from 14 to 493, thus give a diverse test bed. A numeric value in the problem name indicates the number of cities in that tour. As an example, ci76 has 76 cities. A city in a problem is represented as a coordinate; therefore, the TSP cost matrix is prepared calculating distance using the coordinates. As like any TSP solving method, the algorithms of this study took the cost matrix as input of a problem and give sequence of cities as outcome those to be visited in order to make the tour with minimal cost.

Every algorithm requires appropriate parameter settings to get proper outcome. For STPSO and ESTPSO, the value of scaling factor \( \omega \) of Eq. (9) can be calculated using Eq. (11) that is identified to give better result according to the previous study [20].

\[
\omega = 0.1 - \left(1 - \sqrt{\left(1 - 2^{*t / T})\right)}\right) * 0.05
\]

(11)

where \( t \) and \( T \) are current iteration and iteration as termination criteria, respectively. Moreover, the values of \( c_1 \) and \( c_2 \) (i.e., scaling factors) of Eq. (9) were 0.08 and 0.12, respectively as of previous studies. On the other hand, SSPSO and VTPSO does not require any parameter to set for velocity SS calculation. In ACO, alpha and beta were set to 1 and 3, respectively. Population size, i.e., number of particles in the swarm, is a common parameter in any PSO based method. The population size was 100 in all four PSO based methods. Whereas, population size was equal to number of cities in ACO as it desire. On the other hand, fixed number of iteration was considered as the termination criteria of the algorithms and it was set 500 for fair comparison. The selected parameters are not optimal values, but considered for simplicity as well as for fairness in observation.

The algorithms have been implemented on Visual C++ of Visual Studio 2013. The experiments have been conducted on a PC (Intel Core i7-4790 CPU @ 3.60 GHz CPU, 8 GB RAM) with Windows 7 64-bit OS.

**Experimental Results**

This section compares PSO based methods including ACO on the basis of experimental results in solving the benchmark TSPs. Results are presented in two different tables for small and large sized problems. Table 1 presents average and minimum (i.e., best) tour costs achieved by the methods for 20 individual runs for small sized problems. For a particular problem, the best one (i.e., smallest value) among the five 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 summary of the results presented for individual problems. Best/worst summary indicates on how many problem instances a method gave best/worst result among the five methods. Pair wise Win/Draw/Loss summary is for comparing a method with other methods individually. ACO is the prominent method for solving TSP and starts placing an ant in each of individual cities. Thus, it considers population size as the number of cities in a given problem. The tour costs achieved by ACO in different runs are found consistent showing lower Standard Deviation (SD) values. For several problems, especially small sized problems (e.g., ulysses16, gr17, gr21), ACO has shown same tour cost in all 20 individual runs and therefore SD of average tour cost is shown as zero in the Table 1. On the other hand, SSPSO gave most variant outcomes among different runs showing largest SD value for any problem. Among the PSO based methods, ESTPSO has shown the most consistent outcome (i.e., SD value is minimum) and for very small sized problem it showed SD value zero such as burma14, ulysses16, gr17 gr21. However, VTPSO and ESTPSO (even SD values larger than ACO) achieved better average tour costs than ACO for several problems. As an example, ACO achieved average tour cost of 747.12 with 8.13 SD for st70 problem. On the other hand, for the same problem ESTPSO and VTPSO achieved tour costs 722.21 (with SD 23.17) and 716.29 (with SD 17.79), respectively. Since outcome of ACO does not change much in different runs and it is unable to work with population size larger than number of cities, a PSO based method might be a good choice to achieve better outcome from several runs varying population size.

The average tour costs from 20 runs for the small sized problems presented in left side of the Table 1 indicate that ESTPSO and VTPSO are better than ACO and SSPSO is inferior to ACO. SSPSO is shown
on average largest tour cost over the problems (i.e., 57702.65) showing worst for all 24 problems. The average tour cost achieved by ACO is 14576.15 and it is better than SSPSO. On the other hand, VTPSO, the latest PSO based method, has shown best average tour cost of 13413.20 and is found best for 14 cases out of 24 cases. ESTPSO is shown best for 12 cases and outperformed ACO. Although STPSO is worse than ACO on the basis of average result (i.e., 17109.40) it is shown best for three cases but ACO is found best for none. The pairwise Win/Draw/Loss summary indicates that ESTPSO and VTPSO are individually better than ACO in 23 cases; and ACO outperformed ESTPSO and VTPSO for rat99 and gr17, respectively. STPSO is found competitive to ACO showing outperformance on each other for 12 cases. In general, STPSO outperformed ACO for small sized problems (e.g., burma14,ulysses16) and ACO is shown better than STPSO for relatively large sized problems (e.g., kroD100, kroE100). Population size was fixed at 100 for STPSO and it was the number of cities in ACO, therefore, the larger population for relatively large problems in ACO might be the reason for better performance of ACO over STPSO for such large cases. But the outperformance of ESTPSO over STPSO as well as ACO indicates the effectiveness of block node adjustment in ESTPSO. ESTPSO is found better than STPSO for 21 cases and both showed similar outcomes in remaining three cases. On the other hand, VTPSO, which follows different way of improvement of PSO rather than STPSO and ESTPSO, is shown better than STPSO and ESTPSO for 17 and 12 cases, respectively.

Achieved best tour costs from 20 individual runs presented right side in Table 1 also support the average result of left side and indicate the effectiveness of a method to solve benchmark TSP. On the basis of average results of 24 problems, VTPSO is the best and SSPSO is the worst. VTPSO is shown on average lowest tour cost over all the problems (i.e., 19604.71). The average of minimum tour costs are 20825.95, 29203.18 and 19739.41 for ACO, SSPSO, STPSO and ESTPSO, respectively. On the basis of best/worst summary, VTPSO is shown to achieve best tour with shortest path for 22 cases showing worst for none. ACO and ESTPSO are shown the best for remaining 2 and 10 cases, respectively. From pairwise Win/Draw/Loss summary, VTPSO and ESTPSO are better than ACO for 32 cases. Among PSO based methods, VTPSO and ESTPSO are better than SSPSO and STPSO for all 34 cases. On the other hand, VTPSO outperformed ESTPSO for 23 cases and for remaining 11 cases ESTPSO is shown better than VTPSO. Finally, VTPSO is shown the best method among the tested methods and ESTPSO is also found better than ACO.

Experimental analysis

This section investigates the performance of the methods varying population size and number of iteration. The results presented in Tables 1 and 2 were for the fixed number of population size (=100) and iterations (=500) for all the problems. It is necessary to observe how the methods perform on the variation of both the parameters. The experiments were performed on the same machine mentioned before. Three problems with different sizes were selected for the analysis; the problems are eil51, gr96 and gr137.

Figure 1 shows the achieved tour cost for different population sizes that varied from 10 to 500 while total iteration was fixed at 500. The presented results are the averages for five independent runs. Since ACO uses population size equal to the number of cities, the results presented for ACO were only for different runs with fixed population size for a particular problem. Therefore, ACO has shown almost invariant performance. On the other hand, a PSO based method is found to improve with population size. As an example, for eil51 problem at population size 20, STPSO achieved tour cost of 507.53 and is competitive to ACO which achieved tour cost of 504.6. For the same eil51 problem, STPSO showed best tour cost 467.76 at 400 populations that are much better than ACO. It is notable from the figure that SSPSO, the pioneer PSO based method, is the worst among the methods for any population size and much worse than ACO. On the other hand, ESTPSO and VTPSO are found always better than ACO and competitive to one another. However, VTPSO, the latest PSO based method, seems best among the methods and also showed less variant performance when population varied from 50 to 500. This indicates that VTPSO works well and gives suitable performance with relatively small population size. In VTPSO partial search based velocity implementation might be helpful to deliver better outcome even with small population.

Figure 2 compares the variation of termination criteria (i.e., total iteration) on tour costs among the methods. The number of iterations


Int J Swarm Intel Evol Comput, an open access journal
ISSN: 2090-4908
Volume 5 • Issue 3 • 1000139
problem is also interesting to observe computational effectiveness of
requirement comparison among the methods to solve a particular
Figure 1, SSPSO is the worst among the methods for any iteration value
and 534.84 at iteration 800 and 100, respectively. At a glance, similar to
hand, the best tour cost achieved by ESTPSO and VTPSO are 541.72
that improvement was not significant. As an example, ACO achieved
iteration 50, improved with iteration up to a certain value, and after
varied from 50 to 1000 while population size was fixed at 100. The
Citation: Akhand MAH, Hossain SI, Akter S (2016) A Comparative Study of Prominent Particle Swarm Optimization Based Methods to Solve Traveling
Since the experiments are performed in a single machine, the time
ACO SSPSO STPSO ESTPSO VTPSO ACO SSPSO STPSO ESTPSO VTPSO
Comparison of the experimental results of ACO and PSO based methods to solve small sized (up to 100 cities) benchmark TSPs
Table 1: Comparison of the experimental results of ACO and PSO based methods to solve small sized (up to 100 cities) benchmark TSPs.
varied from 50 to 1000 while population size was fixed at 100. The
presented results are the average for five independent runs. It is seen
from the figure that all the methods show the worst tour costs at iteration 50, improved with iteration up to a certain value, and after
that improvement was not significant. As an example, ACO achieved
best tour cost 576.1 at 600 iteration for gr96 problem. On the other
hand, the best tour cost achieved by ESTPSO and VTPSO are 541.72
and 534.84 at iteration 800 and 100, respectively. At a glance, similar to
Figure 1, SSPSO is the worst among the methods for any iteration value and ESTPSO and VTPSO is always better than ACO.
Since the experiments are performed in a single machine, the time
requirement comparison among the methods to solve a particular
problem is also interesting to observe computational effectiveness of
a method. For a particular problem time requirement depends on the
number of population and the number of iteration. As a sample case,
to solve gr96 problem corresponding time requirements for population
variation of Figure 1b and iteration variation in Figure 2b are presented
in Figure 3. It is notable that shape of time requirement curves is also
Recently, PSO has gained popularity in solving difficult optimization problems and a number of PSO based methods have been investigated to solve TSP. A PSO based method calculates velocity of a particle as a function of its position and the best position found so far. The velocity is then used to update the position of the particle in the search space. PSO has been applied to a wide range of optimization problems, including the Traveling Salesman Problem (TSP), a well-known NP-hard problem in combinatorial optimization.

### Table 2: Comparison of the experimental results of ACO and PSO based methods to solve large sized benchmark TSPs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pairwise Win/Draw/Loss Summary on Average Tour Cost and Minimum Tour Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACO</td>
<td>SSPSO</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Best/</td>
<td>Worst</td>
</tr>
</tbody>
</table>

VTPSO took 228.43 s that is less than required by ACO. Although VTPSO calculates velocity similar to other PSO based methods it achieved faster convergence due to partial search based velocity update. Finally, VTPSO performed as the best tour achieving the least time.

### Conclusion

Recently, PSO has gained popularity in solving difficult optimization problems and a number of PSO based methods have been investigated to solve TSP. A PSO based method calculates velocity of a particle as a function of its position and the best position found so far. The velocity is then used to update the position of the particle in the search space. PSO has been applied to a wide range of optimization problems, including the Traveling Salesman Problem (TSP), a well-known NP-hard problem in combinatorial optimization.

### Table 2: Comparison of the experimental results of ACO and PSO based methods to solve large sized benchmark TSPs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pairwise Win/Draw/Loss Summary on Average Tour Cost and Minimum Tour Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACO</td>
<td>SSPSO</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Best/</td>
<td>Worst</td>
</tr>
</tbody>
</table>

VTPSO took 228.43 s that is less than required by ACO. Although VTPSO calculates velocity similar to other PSO based methods it achieved faster convergence due to partial search based velocity update. Finally, VTPSO performed as the best tour achieving the least time.
Figure 1: Variation effect of population size on tour cost.

(a) el51

(b) gr56

(c) gr137

Figure 2: Variation effect of iteration on tour cost.

(a) el51

(b) gr56

(c) gr137
particle considering its current TSP solution with the personal best and global best solutions and then updates its tour incorporating calculated velocity with its solution. In this study, prominent PSO based methods are studied and their outcomes are compared in solving a large number of benchmark TSPs. The methods differ in velocity calculation as well as velocity implementation. In performance evaluation, ACO is also considered since it is a prominent method for TSP. Experimental results revealed that SSPSO (the pioneer method) is the worst among the PSO based methods and even worse than ACO. STPSO is found competitive to ACO; ESTPSO and VTPSO are found better than ACO. But in case of time requirement, ESTPSO is computationally heavy due to its several operations out of basic PSO operations. On the other hand, VTPSO has shown the most cost effective PSO based method and is better than ACO to solve benchmark TSPs.

References

Figure 3: Variation effect of population and iteration on required time for gr 96 problem.


38. TSPLIB - A library of sample instances for the TSP.