International Journal of Swarm Intelligence and Evolutionary Computation

ISSN: 2090-4908

Akhand et al., Int J Swarm Intel Evol Comput 2016, 5:3
DOI: 10.4172/2090-4908.1000139

Research Article

Open Access

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 description of the popular PSO based methods in a similar fashion will be more understandable with distinct features of individuals. Moreover, experimental results on a common benchmark TSP data set will reveal performance of each method.

The rest of the paper is organized as follows. Section II briefly describes basic PSO and popular PSO based methods for TSP. Section III presents as well as compares experimental results of the PSO based methods along with ACO in solving benchmark TSPs. This paper is concluded in Section IV with few remarks.

PSO and its Popular Variants to Solve TSP

This section first explains basic PSO to solve function optimization and then explains popular PSO variants for TSP.

Particle swarm optimization (PSO)

PSO is a population based optimization technique mimicking social behavior of flocks of birds or schools of fishes [10]. PSO has become a popular method in solving difficult optimization problems. At first, PSO generates a random initial population of particles. Each particle maintains a solution of given problem with three parameters: position, velocity and fitness. At every iteration step, each particle changes its position (i.e., search a new point) based on velocity calculated

***Corresponding author:** Akhand MAH, Department of Computer Science and Engineering, Khulna University of Engineering and Technology Khulna, Bangladesh, Tel: 8801714087205; E-mail: akhand@cse.kuet.ac.bd

Received July 05, 2016; **Accepted** August 25, 2016; **Published** August 31, 2016

Citation: Akhand MAH, Hossain SI, Akter S (2016) A Comparative Study of Prominent Particle Swarm Optimization Based Methods to Solve Traveling Salesman Problem. Int J Swarm Intel Evol Comput 5: 139. doi: 10.4172/2090-4908.1000139

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 Xi={x_{i1}, x_{i2}, x_{i3},……x_{iD}}, Pi=(p_{i1},p_{i2},…p_{iD}) is its previous best position and G=(g1,g2,…gD) is the global best position of entire population, the particle calculates its velocity V_{i1},V_{i2},….V_{iD} according to the following equation.

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

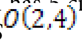
In the equation ω is inertia factor, $c_1$and $c_2$are learning factors, r_1 and r_2 and are vectors of random values between (0,1). After that the particle moves to a new position adding the calculated velocity with its position according to Eq. (2).

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

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_i 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) as mentioned in Step 2. The fitness of the particle is then updated for the new solution $X_i^{(t)}$ and compared with fitness of P_i and G. P_i is updated with $X_i^{(t)}$ if it is found better than P_i (Step 3). Similarly, *G* is updated with $X_i^{(t)}$ if it is found better than *G* (Step 5). In PSO, *G* holds the best solution encountered by the particles throughout the operation; therefore, solution of *G* is considered as the outcome when operation terminates (Step 6). Termination criteria are checked in every iteration after PSO operations (Step 5) and processes (i.e., Step 2 to Step 4) repeat until reaching stopping condition. Usually, a sufficiently good fitness of *G* or a maximum number of iterations is considered as termination criteria.

A number of variations to the basic PSO and incorporation of other techniques with PSO have been investigated to improve its performance [17,19,30-37]. Several variants of PSO is also available to solve TSP [10,21-27], the most studied combinatorial optimization task. To solve TSP, a PSO particle represents a complete TSP tour and velocity associated with it considers a measure to change the tour towards a new tour. The existing studies proposed different techniques and parameters for velocity calculation and hence got new tour. Among the methods, a number of prominent methods use the parameters Swap Operator and Swap Sequence. Following subsections explain the operators in detail and then explain the methods.

Swap operator and swap sequence based operation on TSP

A Swap Operator (SO) contains a pair of indices that indicates two cities may swap in a tour. Suppose, a TSP problem has 5 cities and a solution is *S=(a-c-e-b-d)*. Let a Swap Operator is S then new solution with the SO is like below

S′=S+SO(2,4)=(a-c-e-b-d)+SO(2,4)

=(a-c-e-b-d)

Here, '+' 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:

$$SS_{12}=(SO_1, SO_2, SO_3,…..SO_n), \quad (3)$$

Where $(SO_1, SO_2, SO_3,…..SO_n)$ 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_2 is achieved applying SS_{12} on S_1.

$$S_2=S_1+SS_{12}=S_1+ (SO_1, SO_2, SO_3,……, SO_n) \quad (4)$$

The implication order of SOs of SS_{12} is important because implication of same SOs in different order may give different solutions from the original solution. SS_{12} may also be calculated from solutions S_1 and S_2 in the following equation:

$$SS_{12}=S_2-S_1= (SO_1, SO_2, SO_3,….., SO_n) \quad (5)$$

here '-' means that SOs of SS_{12} need to be applied on solution S_1 to get S_2.

As an example, if S_1=*(a-c-e-b-d)* and S_2=*(b-c-a-e-d)* then SS_{12}=*SO(1,3), SO(2,3), SO(4,5).*

Moreover, 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

$SS(new)=SS_1 \otimes SS_2$

= {*SO* (1,3), *SO* (5,4)} $\otimes$ {*SO* (5,2), *SO* (4,3)} =*SO* (1,3), *SO* (5,4), *SO* (5,2), *SO* (4,3) (6)

It is notable that outcome with different SSs may be same even applying on a solution. Among these SSs which have the least SOs is called Basic Swap Sequence (BSS). As an example, when SSs,

$SS1_{12}$=SO(1,2),SO(2,1),SO(1,3),SO(2,3),SO(4,5) and $SS2_{12}$=SO(1,3), SO(2,3), SO(4,5) is applied on

S_1=*(a-c-e-b-d)* independently, the outcome is S_2=*(b-c-a-e-d)* Therefore, $SS2_{12}$ is the Basic SS. It is also found by using Eq. (5), i.e., S_2–S_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_i) and velocity (V_i) to each individual particle, calculate fitness of each tour, consider P_i as X_i, 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 *($G^{(t-1)}$)* in the population using Eq. (7).

$$V_i^{(t)} = V_i^{(t-1)} \alpha \left(P_i^{(t-1)} - X_i^{(t-1)} \right) \quad \beta \left(G^{(t-1)} - X_i^{(t-1)} \right) \alpha, \beta \grave{o} [1,0] \qquad (7)$$

$$X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)} \qquad (8)$$

$P_i^{(t-1)}$ (and *($G^{(t-1)}$)*) 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 Pi if the new solution $X_i^{(t)}$ is superior to Pi.
Step 4: Update G if the new solution $X_i^{(t)}$ 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), *α, β* are random number between 0 and 1, $\alpha\left(P_i^{(t-1)} - X_i^{(t-1)}\right)$ means all SOs in BSS for $\left(P_i^{(t-1)} - X_i^{(t-1)}\right)$ should be maintained with the probability of α, it is the same for $\beta\left(G^{(t-1)} - X_i^{(t-1)}\right)$. The bigger the value of α (and *β)* the greater the influence of $P_i^{(t-1)}$ (and ($G^{(t-1)}$)) 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 $X_i^{(t)}$ is superior to $P_i$
d. Update $G$ if the new solution $X_i^{(t)}$ is superior to $G$
Step 3: Loop to Step 2 until a termination criterion is met
Step 4: Take the global best solution $G$ as an outcome.
```

After that each particle moves to a new tour ($X_i^{(t)}$) 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 $X_i^{(t)}$ is superior to $P_i$
    d. Update $G$ if the new solution $X_i^{(t)}$ is superior to $G$
Step 3: Tentative Operation on Each Particle $X_i$
    a. Single Node Adjustment
    b. Update $P_i$ if the new solution $X_i^{(t)}$ is superior to $P_i$
    c. 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
```

$$V_i^{(t)} = \omega V_i^{(t-1)} c_1 . r_1 \left(P_i^{(t-1)} - X_i^{(t-1)} \right)$$

$$c_2 . r_2 \left(G^{(t-1)} - X_i^{(t-1)} \right) r_1, r_2 \grave{o} [1,0] \qquad (9)$$

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 scalling factor ω defines the portion of previous veloctiy 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 $X_i^{(t)}$ 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 $X_i$
Single Node Adjustment
Block Node Adjustment
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 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 2 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 α, β 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 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
```

$$V_i^{(t)} = V_i^{(t-1)} \alpha \left(P_i - X_i^{(t-1)}\right) \beta \left(G - X_i^{(t-1)}\right)$$
$$\alpha, \beta \dot{o} [1,0] \qquad (10)$$

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 SOs 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 2.c) 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, eil76 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 ω of Eq. (9) is calculated using Eq. (11) that is identified to give better result according to the previous study [20].

$$\omega = 0.1 - \left(1 - sqr\left(1 - 2 * t / T\right)\right) * 0.05 , \qquad (11)$$

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, respetively 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 of 12803.5. The average minimum tour costs are 14521.8, 54494.97, 15467.3 and 12886.0 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 21 cases showing worst for none. For eil76 problem, as an example, VTPSO achieved best tour path with tour cost of 573.18. For the same problem, tour costs for ACO, SSPSO, STPSO and ESTPSO were 597.95, 2001.76, 674.86 and 586.13, respectively. With similar outcomes for several cases, STPSO and ESTPSO are shown best for 9 and 15 cases, respectively. From pairwise Win/Draw/Loss summary, VTPSO and ESTPSO are better than ACO for all 24 cases. Among PSO based methods VTPSO and ESTPSO are shown better than SSPSO for 24 and 15 cases, respectively. On the other hand, VTPSO outperformed ESTPSO for 9 cases, showed similar outcomes for 12 cases and for rest three cases it was inferior to ESTPSO.

Table 2 presents average and minimum (i.e., best) tour costs achieved by the methods for 20 individual runs for 34 large sized (cities larger than 100) problems. Similar to Table 1, 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 also shows the summary of the results. On the basis of average tour costs, placed at left side of the table, VTPSO is the best and SSPSO is the worst among the five methods. SSPSO is shown worst for all 34 problems and VTPSO is shown best for 25 cases. In such large problems ACO performed best for six cases although it failed to perform best for any small problems (as seen in Table 1). The average tour cost achieved by ACO for all 34 problems is 21065.67; and the value is better than SSPSO and STPSO which showed average tour costs of 217008.2 and 37834.28, respectively. Among PSO based methods, ESTPSO is competitive to ACO with average tour cost of 21042.42 and VTPSO is shown the best method achieving least average tour cost of 20802.25. The pairwise Win/Draw/Loss summary indicates that ESTPSO and VTPSO are better than ACO for 24 and 28 cases, respectively. On the other hand, SSPSO and STPSO are found inferior to ACO for all 34 cases. For such large problems, the better performance of ACO is logical because in such cases population size of ACO was larger than fixed 100 of SSPSO and STPSO. Again, the outperformance of ESTPSO and VTPSO over ACO for 24 and 28 cases indicates the effectiveness block node adjustment in ESTPSO and effectiveness of VTPSO technique.

Achieved best tour costs from 20 individual runs for the large problems presented at the right side in Table 2 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, 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, 205410.5, 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

Sl	Problem	Average Tour Cost (Standard Deviation)										Minimum Tour Cost				
		ACO		SSPSO		STPSO		ESTPSO		VTPSO		ACO	SSPSO	STPSO	ESTPSO	VTPSO
1	burma14	31.31	(0.24)	34.61	(1.16)	**30.87**	(0)	**30.87**	(0)	**30.87**	(0)	31.21	31.84	**30.87**	**30.87**	**30.87**
2	ulysses16	77.13	(0)	83.94	(3.69)	**73.99**	(0)	**73.99**	(0)	74	(0)	77.13	77.36	**73.99**	**73.99**	**73.99**
3	gr17	2332.58	(0)	3215.53	(260.48)	**2332.58**	(0)	**2332.58**	(0)	2342.84	(20.9)	2332.58	2772.53	**2332.58**	**2332.58**	**2332.58**
4	gr21	2955.42	(0)	4958.37	(256.77)	2685.32	(31.15)	**2672.27**	(0)	2697.59	(39.61)	2955.42	4465.08	**2672.27**	**2672.27**	**2672.27**
5	ulysses22	86.18	(0.15)	110.19	(4.59)	75.6	(0.29)	75.36	(0.09)	**75.35**	(0.08)	85.59	99.88	**75.31**	**75.31**	**75.31**
6	gr24	1267.13	(0)	2295.48	(105.69)	1251.59	(7.73)	**1249.82**	(0)	**1249.82**	(0)	1267.13	2021.14	**1249.82**	**1249.82**	**1249.82**
7	fri26	646.39	(0.37)	1229.12	(73.02)	636.62	(4.52)	**635.58**	(0)	637.41	(4.94)	644.8	1089.92	**635.58**	**635.58**	**635.58**
8	bayg29	9964.78	(0)	17427.38	(919.49)	9102.34	(82.61)	**9074.15**	(0)	9133.78	(103.12)	9964.78	14914.93	**9074.15**	**9074.15**	**9074.15**
9	bays29	9964.78	(0)	17777.28	(655.35)	9091.49	(42.15)	**9074.15**	(0)	9092.9	(52.32)	9964.78	16092.24	**9074.15**	**9074.15**	**9074.15**
10	hk48	12707.6	(11.88)	34555.42	(1269.5)	12568.62	(694.69)	**11125.15**	(52.41)	11551.63	(303.42)	12699.86	31607.75	11616.3	**11104.67**	**11104.67**
11	att48	38989.37	(0)	108035.27	(3646.6)	37040.66	(1928.7)	**33892.75**	(183.5)	34561.61	(553.46)	38989.37	99784.19	34076.62	**33784.02**	**33784.02**
12	eil51	500.87	(7.14)	1235.1	(33.63)	490.07	(16.59)	445.43	(4.63)	**444.57**	(5.86)	473.11	1149.68	460.39	435.62	**435.35**
13	berlin52	8074.63	(31.59)	22208.33	(595.27)	8542.19	(360.55)	**7761.66**	(170.1)	7875.94	(216.04)	7966.99	20294.9	7684.23	**7544.37**	**7544.37**
14	st70	747.12	(8.13)	2799.86	(81.36)	903.53	(54.52)	722.21	(23.17)	**716.29**	(17.79)	734.19	2621.95	810.59	**687.17**	687.83
15	eil76	597.95	(7.03)	2001.76	(41.07)	674.86	(23.78)	586.13	(10.63)	**573.18**	(7.38)	580.91	1925.04	617.31	569.25	**562.94**
16	pr76	127371.7	(0)	451769.47	(9267.9)	144306.46	(7518.8)	**113158.94**	(3377.9)	113603.24	(2860.9)	127371.68	433409.25	132607.74	109059.3	**108981.2**
17	gr96	587.96	(8.6)	2702.02	(51.52)	725.8	(43.14)	554.35	(26.52)	**544.7**	(14.28)	567.52	2595.8	643.7	**516.1**	523.07
18	rat99	1369.2	(0.85)	6571.79	(139.97)	1750.38	(107.56)	1389.78	(47.04)	**1345.47**	(32.05)	1366.3	6260.14	1475.11	1295.55	**1261.13**
19	kroa100	24645.72	(73.91)	134460.78	(2742.0)	33198.45	(1724.5)	23506.66	(860.23)	**22214.29**	507.14)	24524.53	128768.75	29043.03	21622.39	**21408.48**
20	kroB100	25234.1	(481.4)	131988.83	(3281.7)	32542.56	(2272.9)	24393.93	(619.48)	**23314.96**	506.74)	24675.03	122797	28137.74	23236.43	**22367.61**
21	kroC100	23324.62	(131.3)	132512.47	(3337.7)	32537.78	(2089.0)	22387.53	(762.92)	**22196.17**	513.44)	23248.13	123441.85	29969	21347.21	**21268.17**
22	kroD100	24399.57	(15.5)	127337.76	(3974.8)	34642.48	(2113.8)	23508.77	(1228.5)	**22623.57**	552.35)	24396.01	120331.25	30699.7	**21547.04**	21650.61
23	kroE100	24530.33	(157.8)	135175.79	(2772.8)	33389.21	(2838.3)	24407.89	(792.4)	**23466.54**	598.72)	24396.38	130475.25	27945.66	23111.2	**22516.1**
24	rd100	9421.11	(60.17)	44377.02	(1271.6)	12032.2	(734.04)	8856.77	(361.24)	**8557**	265.87)	9210.67	40851.51	10209.74	8186.36	**7971.41**
	Average	**14576.15**		**57702.65**		**17109.40**		**13413.20**		**13288.49**		**14521.84**	**54494.97**	**15467.32**	12886.06	**12803.57**
	Best/ Worst	**0/0**		**0/24**		**3/0**		**12/0**		**14/0**		**0/0**	**0/24**	**9/0**	**15/0**	**21/0**

Method	**Pairwise Win/Draw/Loss Summary on Average Tour Cost and Minimum Tour Cost**									
	ACO	**SSPSO**	**STPSO**	**ESTPSO**	**VTPSO**	**ACO**	**SSPSO**	**STPSO**	**ESTPSO**	**VTPSO**
ACO	-	0/0/24	12/0/12	23/0/1	23/0/1	-	0/0/24	13/0/11	24/0/0	24/0/0
SSPSO		-	24/0/0	24/0/0	24/0/0		-	24/0/0	24/0/0	24/0/0
STPSO			-	21/3/0	17/1/6			-	15/9/0	15/9/0
ESTPSO					12/2/10				-	9/12/3

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 similar for eil51 and gr137 problems. It is observed from the figure that STPSO took much time than SSPSO because it introduced single node adjustment over SSPSO and ESTPSO took much time than STPSO because it introduced block node adjustment upon STPSO. On the other hand, ACO took less time than SSPSO; and VTPSO is competitive to ACO. As an example, for 500 iterations SSPSO, STPSO and ESTPSO took 346.55, 420.33 and 600.0 s, respectively. For the same 500 iterations, ACO took 249.88 s. Among the PSO based methods,

Sl	Problem	Average Tour Cost (Standard Deviation)										Minimum Tour Cost				
		ACO		SSPSO		STPSO		ESTPSO		VTPSO		ACO	SSPSO	STPSO	ESTPSO	VTPSO
1	eil101	737.44	(6.64)	2799.6	(55.71)	841.14	(54.15)	696.12	(11.97)	**841.14**	(9.92)	715.18	2638.47	753.94	672.75	**655.32**
2	lin105	**15522.5**	(316.84)	96919.84	(2316.55)	23162.54	(2384.2)	16114.34	(790.08)	23162.54	(520.81)	15364.58	91743.07	19359.05	15085.39	**14701.58**
3	pr107	46557.69	116.01)	439116.3	(8574.2)	79486.49	(10357)	**44632.63**	(152.47)	79486.49	(1196.7)	46413.52	423469.7	53122.02	**44301.7**	44584.38
4	pr124	65145.25	(0)	565621.5	(10300.14)	100979	(11571.8)	66481.58	(2618)	**100979**	(1094.0)	65145.25	541989.1	79706.86	62501.23	**61662.91**
5	bier127	128108.2	(1239.1)	532280.7	(11315.41)	161793.5	(9365.7)	126838	(2872.1)	**161793.5**	(2398.0)	124559.4	500526.7	148245.8	**121039.1**	121082.5
6	ch130	7043.3	(79.38)	38272.61	(1007.29)	9563.97	(630.88)	6733.25	(163.74)	**9563.97**	(155.1)	6917.91	35046.95	8538.29	6444.42	**6215.44**
7	pr136	110876	(16.33)	681912.2	(15325.97)	153750.3	(10588.1)	106603.1	(4022.4)	**153750.3**	(2461.8)	110872.2	625531.6	127394.6	**99219.27**	99942.05
8	gr137	926.85	(8.95)	4938.19	(154.07)	1167.4	(108.41)	809.29	(18.95)	**1167.4**	(17.61)	899.09	4596.79	1008.87	760.18	**744.18**
9	pr144	**59627.42**	(182.34)	671970.6	(13867.41)	136396.1	(12809.8)	66097.9	(2920.6)	136396.1	(2536.2)	**59415.4**	635296.6	114332.8	59547.19	59647.36
10	ch150	**6907.15**	(73.31)	45323.59	(711.77)	10344.47	(753.51)	7388.19	(208.37)	10344.47	(162.49)	**6770.18**	44110.16	8753.37	7028.03	6852.18
11	kroA150	30759.58	(317.35)	212961.4	(3320.12)	45118.03	(4028.44)	29300.59	(520.57)	**45118.03**	(583.37)	30040.62	204164.8	40186.03	28526.15	**27212.81**
12	kroB150	31053.56	(162.19)	209840.5	(3584.08)	44153.4	(2972.24)	28998.75	(726.54)	**44153.4**	(631.66)	30604.29	197637.7	38407.87	27885.29	**26776.94**
13	pr152	79504.02	(181.38)	880786.2	(8982.42)	154613.9	(15284.8)	79432.29	(2062.2)	**154613.9**	(1507.6)	79153.02	857622.4	128456.6	76726	**74107.38**
14	u159	47793.27	(309.99)	372243.5	(8821.78)	71030.94	(7560.71)	47970.1	(1431.5)	**71030.94**	(1383.2)	47514.43	347189.8	56732.94	45354.17	**44083.4**
15	rat195	**2570.99**	(27.35)	19296.1	(268.74)	3781.4	(307.53)	2657.21	(76.82)	3781.4	(49.58)	2534.75	18623.87	3223.31	2508.52	**2473.74**
16	d198	17456.37	(146.45)	155376.2	(2762.86)	29275.75	(2765.72)	17308.64	(171.01)	**29275.75**	(239.69)	17301.47	150029	23170.7	17017.81	**16104.26**
17	kroA200	34620.74	(174.42)	286928	(5161.32)	54925.81	(5570.29)	32648.93	(862.85)	**54925.81**	(520.9)	34547.69	273295.2	43842.92	31304.88	**30707.25**
18	kroB200	35127.96	(344.72)	281184.7	(6593.01)	51218.46	(4437.39)	32710.35	(628.33)	**51218.46**	(717.11)	34207.79	261440.5	44143.71	31554.36	**30514.27**
19	gr202	569.97	(2.1)	2747.51	(50.37)	708.17	(35.06)	528.3	(8.73)	**708.17**	(7.47)	566.08	2602.78	653.21	512.17	**504.62**
20	ts225	**137421**	(62.63)	1386139	(15612.99)	203659.1	(15882.4)	142049.5	(4738.6)	203659.1	(3344.5)	137358.4	1337783	175849.7	**132591.6**	134505.6
21	tsp225	4547.87	(59.27)	35850.1	(498.44)	6094.45	(401.84)	4356.25	(98.76)	**6094.45**	(63.08)	4472.68	34613.2	5352.26	4179.77	**4139.85**
22	pr226	90550.08	(145.82)	1461647	(17035.9)	179124.4	(19713.2)	93182.75	(3927.4)	**179124.4**	(3025.7)	90501.46	1422765	127690	**83840.3**	84343.33
23	gr229	1925.31	(22.14)	13639.65	(320.98)	2494.74	(241.34)	1784.97	(33.21)	**2494.74**	(24.73)	1865.63	12644.35	2112.95	1720.43	**1712.69**
24	gil262	2796.2	(22.13)	23616.51	(262.73)	4238.6	(251.12)	2673.61	(57.12)	**4238.6**	(33.04)	2767.35	22813.56	3801.81	2578.68	**2577.09**
25	pr264	**54388.15**	(121.6)	938471.6	(17197.82)	130455.7	(10701.2	55732.53	(1123.0)	130455.7	(978.38)	54206.21	904252.3	110299	**53459.95**	53766.16
26	pr299	57205.3	(361.35)	653271.6	(14348.11)	92542.76	(8643.73)	56468.8	(11355)	**92542.76**	(953.62)	57051.19	615387.7	74224.96	54614.16	**52069.8**
27	lin318	48511.98	(319.68)	522028.6	(6432.04)	86710.79	(4269.7)	47532.54	(1011.6)	**86710.79**	(651.62)	48126.45	506435.5	78735.93	45705.48	**45284.7**
28	linhp318	48392.22	(355.54)	524957.8	(6525.9)	84472.38	(4819.54)	47708.68	(983.28)	**84472.38**	(722.67)	47984.96	503906.9	77181.54	46149.11	**45575.61**
29	rd400	17921.84	(118.08)	191166.5	(1426.84)	27744.46	(1639.29)	**16890.71**	(276.56)	27744.46	(204.3)	17754.66	186799.9	24943.48	**16361.51**	16589.19
30	fl417	13554.67	(111.29)	437355.2	(2844.06)	33195.24	(3983.01)	13740.95	(446.37)	**33195.24**	(337.72)	13390.96	431821.1	26455.5	12964.52	**12409.94**
31	gr431	2373.62	(19.74)	25599.18	(365.42)	3475.87	(197.85)	2121.45	(38.51)	**3475.87**	(24.81)	2336.38	24819.06	3043.36	**2020.22**	2021.43
32	pcb442	58469.86	(108.89)	703108.9	(7123.31)	88675.92	(5954.78)	57322	(1339.6)	**88675.92**	(928.78)	58288.81	678325.6	77460.67	**53923.08**	54956.45
33	pr439	127423.3	(312.16)	1721145	(13467.78)	237200.6	(18261.1)	**121396**	(3551.0)	237200.6	(2131.5)	127228.3	1677427	204236.6	**116092**	117113.8
34	d493	39807.93	(422.63)	406682.7	(4723.98)	60717	(3072.74)	38992.88	(434.14)	**60717**	(652.84)	39152.43	390534.4	54361.93	38054.69	**37448.02**
	Average	**21065.67**		**217008.2**		**37834.28**		**21402.42**		**20802.25**		**20825.95**	**205410.5**	**29203.18**	**19739.41**	**19604.71**
	Best/ Worst	**6/0**		**0/34**		**0/0**		**3/0**		**25/0**		**2/0**	**0/34**	**0/0**	**10/0**	**22/0**

Method	Pairwise Win/Draw/Loss Summary on Average Tour Cost and Minimum Tour Cost									
	ACO	SSPSO	STPSO	ESTPSO	VTPSO	ACO	SSPSO	STPSO	ESTPSOO	VTPSO
ACO	-	0/0/34	0/0/34	24/0/10	28/0/6	-	0/0/34	0/0/34	32/0/2	32/0/2
SSPSO		-	34/0/0	34/0/0	34/0/0		-	34/0/0	34/0/0	34/0/0
STPSO			-	34/0/0	34/0/0			-	34/0/0	34/0/0
ESTPSO				-	31/0/3				-	23/0/11

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

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 implementation. Finally, VTPSO performed as the best tour achieving method using 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

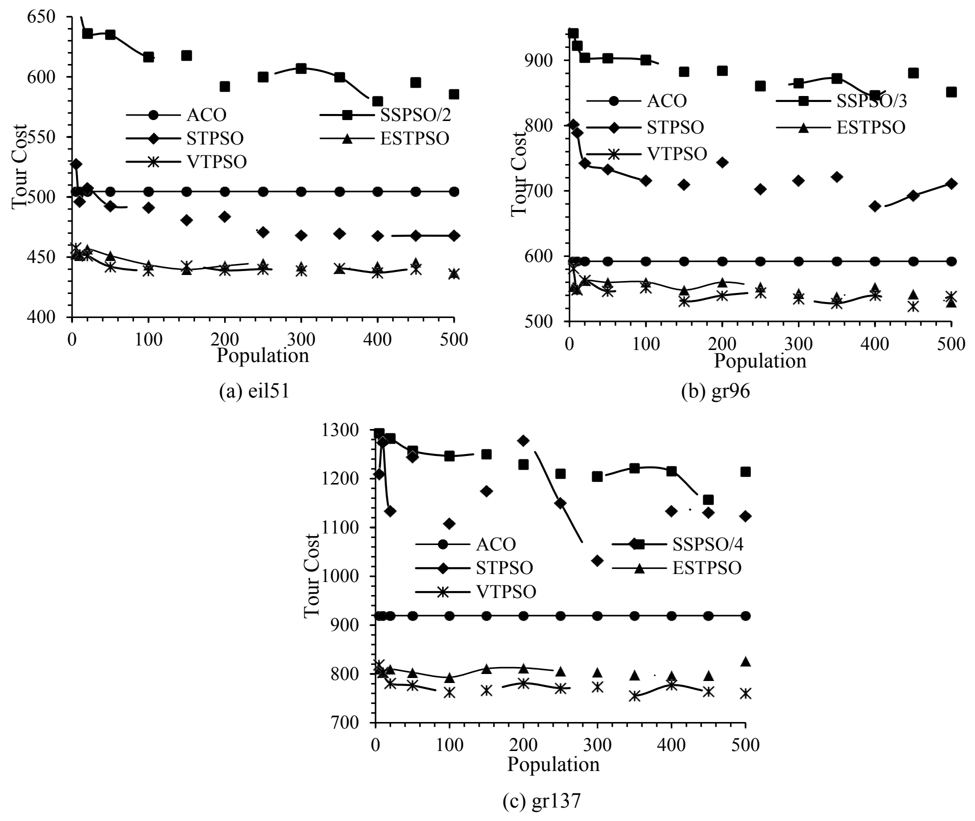


Figure 1: Variation effect of population size on tour cost.

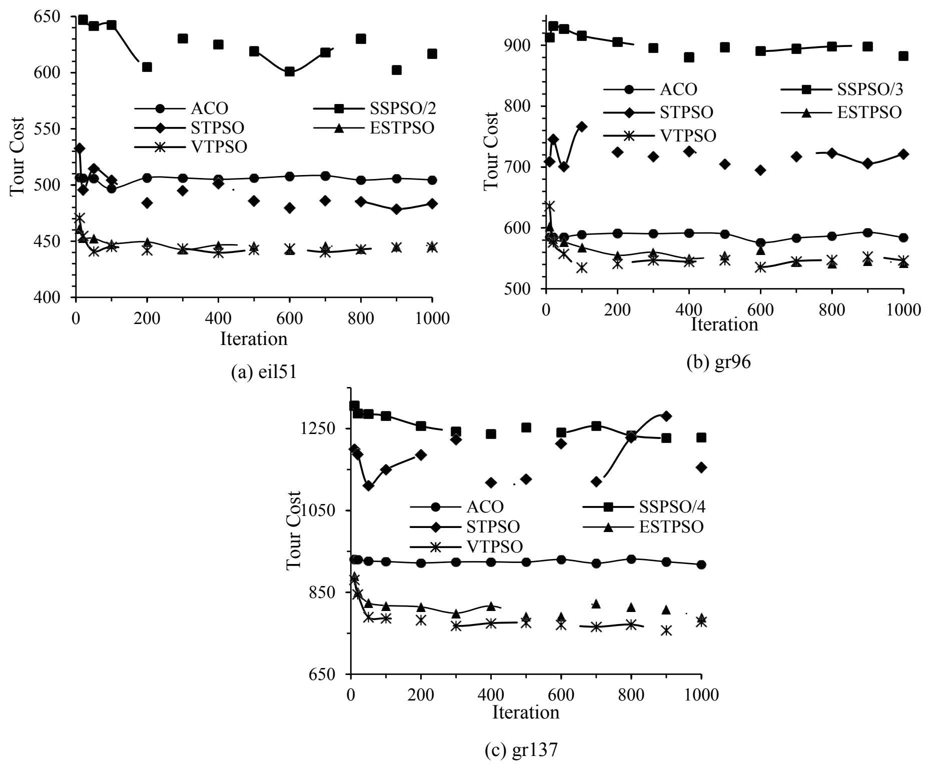


Figure 2: Variation effect of iteration on tour cost.

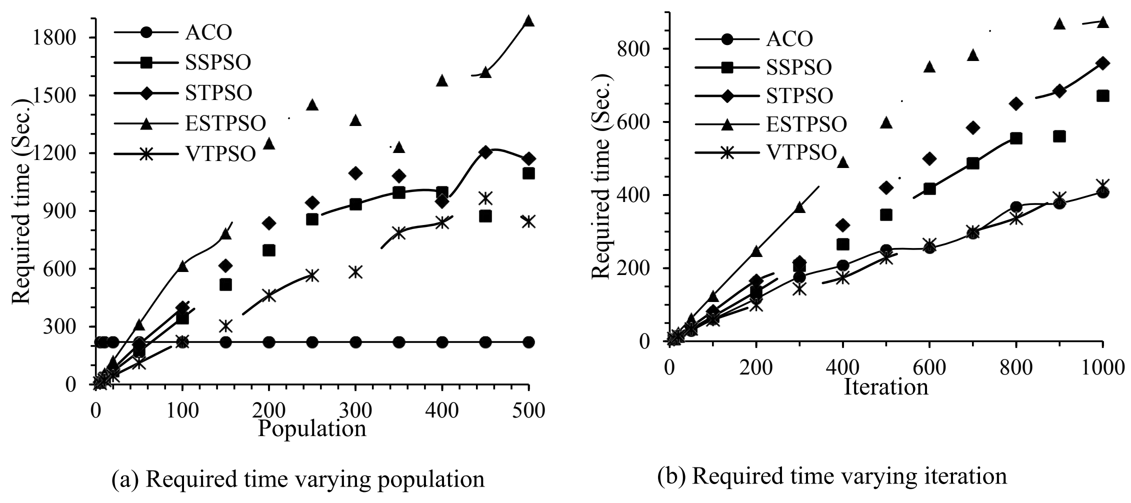


Figure 3: Variation effect of population and iteration on required time for gr 96 problem.

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 reveled 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

1. Goldberg DE (1998) Genetic algorithms. Addison-wesley.
2. Whitely D (1994) A genetic algorithm tutorial. Statistics and Computing 4: 65-85.
3. Vasant P, Barsoum N (2009) Hybrid genetic algorithms and line search method for industrial production planning with non-linear fitness function. Engineering Applications of Artificial Intelligence 22: 767-777.
4. Senvar O, Turanoglu E, Kahraman C (2012) Usage of metaheuristics in engineering: A literature review. In engineering, business, economics and finance pp: 484-528.
5. Brownlee J (2011) Clever algorithms: Nature-inspired programming recipes.
6. Pal SK, Rai CS, Singh AP (2012) Comparative study of firefly algorithm and particle swarm optimization for noisy non-linear optimization problems. International Journal of Intelligent Systems and Applications (IJISA) 4: 50-57.
7. Khan K, Sahai A (2012) A glow worm optimization method for the design of web services. International Journal of Intelligent Systems and Applications 10: 89-102.
8. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: From natural to artificial systems. Oxford University Press, Oxford.
9. Cordon O, Herrera F, Stutzle T (2002) A review on the ant colony optimization metaheuristic: Basis, models and new trends. Mathware and Soft Computing 9: 141-175.
10. Eberhart R, Kennedy J (1995) A new optimizer using particles swarm theory. Roc Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan) IEEE Service Center, Piscataway NJ: 39-43.
11. Wang KP, Huang L, Zhou CG, Pang W (2003) Particle swarm optimization for traveling salesman problem. International conference on machine learning and cybernetics, Xi'an pp: 1583-1585.
12. Wong L, Low MYH, Chong CS (2008) A bee colony optimization algorithm for traveling salesman problem. Second Asia International Conference on Modeling & Simulation p: 8.
13. Yuce B (2014) The bees algorithm and its applications. Handbook of research on artificial intelligence techniques and algorithms pp: 122-151.
14. Vora M, Mirnalinee TT (2014) From optimization to clustering: A swarm intelligence approach. Handbook of research on artificial intelligence techniques and algorithms pp: 594-619.
15. Akhand MAH, Shill PC, Hossen MF, Junaed ABM, Murase K (2015) Producer-Scrounger method to solve traveling salesman problem. International Journal of Intelligent Systems and Applications 7: 29-36.
16. He S, Wu QH, Saunders JR (2009) Group search optimizer: An optimization algorithm inspired by animal searching behaviour. IEEE transactions on evolutionary computation 13: 973-990.
17. Zhang L, Tian F, Kadri C, Pei G, Li H, Pan L (2011) Gases concentration estimation using heuristics and bio inspired optimization models for experimental chemical electronic nose. Sensors and actuators B: Chemical 160: 760-770.
18. Assadi MK, Zahraee SM, Taghdisi J (2016) Integration of computer simulation, design of experiments and particle swarm optimization to optimize the production line efficiency. International Journal of Swarm Intelligence and Evolutionary Computation 5: 4.
19. Mange J, Pace S (2016) Scheduling functions for position updating in population based optimization algorithms. International Journal of Swarm Intelligence and Evolutionary Computation 5: 133.
20. Yan X, Zhang C, Luo W, Li W, Chen W, Liu H (2012) Solve traveling salesman problem using particle swarm optimization algorithm. International Journal of Computer Science Issues 9: 264-271.
21. Wei X, Jiang-wei Z, Hon-lin Z (2009) Enhanced self-tentative particle swarm optimization algorithm for TSP. Journal of North China Electric Power University 36: 69-74.
22. Zhang J, Si W (2010) Improved enhanced self-tentative PSO algorithm for TSP. Proc. of sixth IEEE International Conference on Natural Computation pp: 2638-2641.
23. Shi XH, Liang YC, Lee HP, Lu C, Wang QX (2007) Particle swarm optimization-based algorithms for TSP and generalized TSP. Information Processing Letters 103: 169-176.
24. Fan H (2010) Discrete particle swarm optimization for TSP based on neighborhood. Journal of Computational Information Systems (JCIS) 6: 3407-3414.

25. Goldbarg EFG, Goldbarg MC, de Souza GR (2008) Particle swarm optimization algorithm for traveling salesman problem, traveling salesman problem. Federico Greco (Ed.) Intec.

26. Akhand MAH, Akter S, Rashid MA, Yaakob SB (2015) Velocity tentative PSO: An optimal velocity implementation based particle swarm optimization to solve traveling salesman problem. IAENG international journal of computer science 42: 221-232.

27. Montero E, Riff MC, Altamirano L (2011) A PSO algorithm to solve a real course+exam timetabling problem. International conference on swarm intelligence, Cergy, France pp: 14-15.

28. Matai R, Singh SP, Mittal ML (2010) Traveling salesman problem: An overview of applications, formulations, and solution approaches, traveling salesman problem, theory and applications. InTech pp: 1-24.

29. Abduljabbar ZA, Khalefa MS, Jabar MA (2013) Comprison between ant colony and genetic algorithm using travelling salesman problem. International Journal of Soft Computing 8: 171-174.

30. Liao YF, Yau DH, Chen CL (2012) Evolutionary algorithm to traveling salesman problems. International Journal of Computers & Mathematics with Applications 64: 788-797.

31. Marinakisa Y, Marinaki (2010) A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. Journal of computers & operations research, Elsevier Publisher pp: 432-442.

32. Rosendo M, Pozo A (2010) A hybrid particle swarm optimization algorithm for combinatorial optimization problems. IEEE congress the conference on evolutionary computation pp: 18-23.

33. Vo DN, Schegner P (2012) An improved particle swarm optimization for optimal power flow in meta-heuristics optimization algorithms in engineering, business, economics and finance pp: 1-40.

34. Popa R (2013) Melanocytic lesions screening through particle swarm optimization. Handbook of research on novel soft computing intelligent algorithms: Theory and practical applications pp: 355-384.

35. Garg H, Rani M, Sharma SP (2013) Predicting uncertain behavior and performance analysis of the pulping system in a paper industry using pso and fuzzy methodology. Handbook of research on novel soft computing intelligent algorithms: Theory and practical applications pp: 414-449.

36. Majumder AR, Majumder AB (2014) Application of standard deviation method integrated pso approach in optimization of manufacturing process parameters. Handbook of research on artificial intelligence techniques and algorithms pp: 536-563.

37. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. Journal of evolutionary computation 10: 281-295.

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