Vietnam J Comput Sci (2018) 5:27–43
https://doi.org/10.1007/s40595-017-0099-z

REGULAR PAPER

A novel hybrid algorithm for generalized traveling salesman problems in different environments

Indadul Khan[1] · **Manas Kumar Maiti**[2]

Received: 1 December 2016 / Accepted: 11 July 2017 / Published online: 21 July 2017

Abstract A swap sequence-based particle swarm optimization (SSPSO) technique and genetic algorithm (GA) are used in tandem to develop a hybrid algorithm to solve generalized traveling salesman problem. Local search algorithm K-Opt is occasionally used to move any stagnant solution. Here, SSPSO is used to find the sequence of groups of a solution in which a tour to be made and cities from different groups of the sequence are selected using GA. The K-Opt algorithm (for $K = 3$) is used periodically for a predefined number of iterations to improve the quality of the solutions. The algorithm is capable of solving the problem in crisp as well as in imprecise environment. For this purpose, a general fitness evaluation rule for the solutions is proposed. The efficiency of the algorithm is tested in crisp environment using different size benchmark problems available in TSPLIB. In crisp environment, the algorithm gives 100% success rate for problems up to considerably large sizes. Imprecise problems are generated from crisp problems randomly using a rule and are solved using the proposed approach. The obtained results are discussed. Moreover it is observed that the proposed algorithm finds multiple optimal paths, when they exist, both for the crisp problems and their fuzzy variations.

Keywords Traveling salesmen problem · Genetic algorithm · Particle swarm optimization · Swap sequence · Swap operation · K-Opt

1 Introduction

The traveling salesmen problem (TSP) is one of the standard combinatorial discrete optimization problems. The generalized traveling salesman problem (GTSP) is an extension of TSP and is a NP-hard problem. The GTSP has been introduced by [14], Saksena [31], and Srivastava [33] in the context of computer record balancing and of visit sequencing through welfare agencies since the 1960s. The problem consists of a set of n cities where travel cost c_{ij} between (i-th city and j-th city) any two cities is known. The n cities are divided into several groups, i.e., $g_1, g_2, \ldots, g_m$. Here, m is the numbers of groups. A city may belong to one or more than one group. The objective is to find the possible minimum-cost Hamiltonian cycle (tour) through the groups. The GTSP has various real-world applications such as mail delivery [19], welfare agency routing ([31], material flow system design [19], vehicle routing [19], and computer file sequencing [14]. The existing algorithms for GTSP are mainly based on dynamic programming techniques [8,14,18,26,31,33], which transfer GTSP into TSP. Some heuristic and meta-heuristic algorithms have been proposed for solving GTSP: a random key genetic algorithm [34], a memetic algorithm [13], an efficient composite heuristic [30], reinforcing ant colony system [27], etc. Yang et al.[37] present a new heuristic method-based ant colony optimization (ACO) for solving GTSP. Since 1995, particle swram optimization (PSO) [16] has been proven to succeed in continuous optimization problems, and many works have been done effectively in this area. PSO can be used to solve

✉ Indadul Khan
indadulkhan@gmail.com

Manas Kumar Maiti
manasmaiti@yahoo.co.in

[1] Department of Computer Science, Chandrakona Vidyasagar Mahavidyalaya, Paschim-Medinipur, West Bengal 721201, India

[2] Department of Mathematics, Mahishadal Raj College, Mahishadal, Purba-Medinipur, West Bengal 721628, India

GTSP also. Using the concept of the swap operator and swap sequence, and redefining some operators of PSO on the basis of them, Shi et al. [32] present another heuristic method based on PSO for solving TSP as well as GTSP. On the other hand, the genetic algorithm (GA) was introduced by Holland, inspired by Darwin's theory in 1970s [15]. The idea behind GA is to model the natural evolution by using genetic inheritance together with Darwin's theory. Nowadays, GA is one of the important heuristic algorithms for NP-hard combinatorial optimization problems. Bontouxa et al. [3] propose a memetic algorithm using GA for solving GTSP. Zhao and Zhu [40] present an innovative GA-based algorithm for solving GTSP.

In all the above literature, it is implicitly assumed that the traveling cost from one city to another is fixed, i.e., crisp in nature. Traveling cost from one city to another depends on the conveyance used for traveling. It varies slightly depending on the availability of the conveyance, condition of the road, etc., though its value normally lies in an interval. So, these costs are imprecise in nature. There are different estimation approaches for imprecise data. Among them, stochastic estimation [22], fuzzy estimation [5,12,17] and rough estimation [24] have drawn more attention. For real-life problem like GTSP, fuzzy estimation of travel costs is more appropriate. It is less error prone, as these estimations are based on experts' opinion. Also, not much past data are required for such an estimation. Due to this reason, it is better to model the costs of a GTSP as fuzzy numbers. Though there are some research works on TSP incorporating fuzzy costs [4,5,17], none has solved GTSP in a fuzzy environment.

From the above discussion, it is clear that there are some lacunas in the existing literature of GTSP which are summarized below:

- In the existing literature, GTSPs are considered only in a crisp environment.
- Existing algorithms for GTSP are not capable of solving GTSPs in an imprecise environment.
- The presence of multiple paths of any GTSP is overlooked/ignored by the researchers.

To remove the above-mentioned shortcomings in this research paper, a GTSP is considered whose costs are fuzzy in nature. To solve this problem, an algorithm is proposed which is capable of solving a GTSP in crisp environment as well as in fuzzy environment. The proposed algorithm is a hybridization of SSPSO, GA and K-Opt. Initially, a sequence of groups in which a tour is to be made is selected randomly and for each sequence a city from each group is selected using GA. After the first iteration, SSPSO is used to rearrange the sequence of groups for improvement of the solution and cities from different groups of the sequence selected using GA. In this GA, two types of crossover operations—single-point crossover and multi-point crossover—are performed depending on some random functions. For mutation operation, an adaptive mutation function is used that continues to perform mutation operation until no better movement is found in a predefined number of iterations. A strong local search algorithm K-Opt ($K = 3$) is used periodically for a predefined number of iterations to improve the quality of the solutions. This predefined number of iterations may vary with the size of the problem. The algorithm is capable of solving the problem in crisp as well as in imprecise environment. A general fitness evaluation scheme is proposed for the purpose. As a result, the algorithm can be used to solve GTSP in fuzzy, rough and stochastic environments. The efficiency of the algorithm is tested in crisp environment using different size benchmark problems available in TSPLIB [29]. In crisp environment, the algorithm gives 100% success rate for problems up to considerably large sizes. Imprecise problems are generated from crisp problems randomly using a proposed rule. Imprecise problems are solved and the obtained results are discussed.

The rest of the paper is organized as follows: in Sect. 2, problem definition and some mathematical prerequisites are presented. In Sect. 3, some features of SSPSO are discussed. The features of GA are discussed in Sect. 4. The K-Opt algorithm is presented in Sect. 5. The proposed algorithm is presented in Sect. 6. The experimental results are discussed in Sect. 7. A brief conclusion is drawn in Sect. 8. At the end, the reference list is presented.

2 Problem definition and mathematical prerequisites

The GTSP can be described as the problem of seeking a special Hamiltonian cycle with the lowest cycle cost in a complete weighted graph. Let $G = \{V, E, C\}$ be a complete weighted graph where $V = \{v_1, v_2, \ldots, v_n\}$ $(n \geq 3)$, $E = \{e_{ij} | v_i, v_j \in V\}$ and $C = \{c_{ij} | c_{ij} > 0$ and $c_{ii} = 0, \forall i, j \in \{1, 2, \ldots, n\}\}$ are a set of cities, set of edges between cities, and the set of travel cost between two cities, respectively. The symbols v_i, e_{ij}, c_{ij} represent the i-th city, the edge connecting cities v_i and v_j, and the travel cost corresponding to edge e_{ij}, respectively. The city set V is partitioned into m possibly intersecting groups $g_1, g_2, \ldots, g_m$ with $|g_j| \geq 1$ and $V = \bigcup_{j=1}^{m} g_j$, where $|g_j|$ is the number of elements in group $|g_j|$. The special Hamiltonian cycle is required to pass through all of the groups, but not all of the cities differing from that of TSP. In real-life problem, c_{ij} represents the cost of travel between city i and city j, or distance between city i and city j, or time required to travel from city i to city j, etc. There are two different kinds of GTSP under the above-mentioned

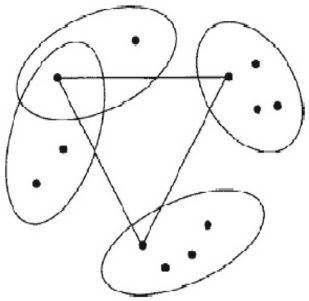

Fig. 1 Tpye-1 GTSP (only one node from each group)

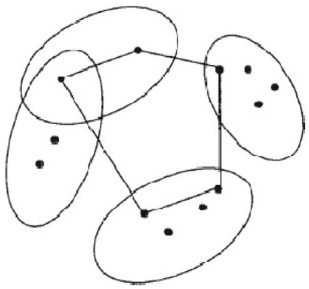

Fig. 2 Type-1 GTSP (At least one node from each group)

framework of the special Hamiltonian cycle [8,19]: (1) the cycle passes through exactly one city in each group (Fig. 1) [14,31,33] and (2) the cycle passes through at least one city in each group (Fig. 2) [18,26]. The first kind of GTSP is known as E-GTSP, where E stands for equality [8]. In this paper only the E-GTSP, i.e., the first case, is discussed and is still called GTSP for convenience. The problem is to find a travel schedule for a salesman who likes to visit exactly one city of each group $(g_1, g_2, \ldots, g_m)$ and comes back to the starting city with minimum tour cost. Here, it is assumed that groups of cities are known, i.e., the salesman will not determine the group.

Fuzzy number A fuzzy number $\tilde{a}$ is a fuzzy set in $\Re$ (set of real numbers) characterized by a membership function, $\mu_{\tilde{a}}(x)$, which is both normal (i.e., $\exists$ at least one point x $\in \Re$ s.t. $\mu_{\tilde{a}}(x) = 1$) and convex [39].

Triangular fuzzy number (TFN) A TFN $\tilde{a} = (a_1, a_2, a_3)$ has three parameters a_1, a_2, a_3, where $a_1 < a_2 < a_3$, whose membership function $\mu_{\tilde{a}}(x)$ is given by

$$\mu_{\tilde{a}}(x) = \begin{cases} \frac{x-a_1}{a_2-a_1} & \text{for } a_1 \le x \le a_2 \\ \frac{a_3-x}{a_3-a_2} & \text{for } a_2 \le x \le a_3 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Possibility (Pos), necessity (Nes) measure Let $\tilde{a}$ and $\tilde{b}$ be two fuzzy numbers with membership functions $\mu_{\tilde{a}}$ and $\mu_{\tilde{b}}$, respectively. Then taking the degree of uncertainty as the semantics of fuzzy number:

$$\text{Pos}(\tilde{a} \star \tilde{b}) = \text{Sup}\{\min(\mu_{\tilde{a}}(x), \mu_{\tilde{b}}(y)), x, y \in \Re,\ x \star y\}, \tag{2}$$

where $\star$ is any one of the relations $>, <, =, \le, \ge$. Analogously, if $\tilde{b}$ is a crisp number, say b, then

$$\text{Pos}(\tilde{a} \star b) = \text{Sup}\{(\mu_{\tilde{a}}(x), x \in \Re, x \star b\}. \tag{3}$$

On the other hand, the necessity measure of an event $\tilde{a} \star \tilde{b}$ is a dual of the possibility measure. The grade of necessity of an event is the grade of impossibility of the opposite event and is defined as

$$\text{Nes}(\tilde{a} \star \tilde{b}) = 1 - \text{Pos}(\overline{\tilde{a} \star \tilde{b}}), \tag{4}$$

where $\overline{\tilde{a} \star \tilde{b}}$ represents the complement of the event $\tilde{a} \star \tilde{b}$.

Credibility (Cr) measure Using (3) and (4), the credibility measure of an event $\tilde{a} \star \tilde{b}$ is denoted by Cr($\tilde{a} \star \tilde{b}$) and is defined as [21]

$$\text{Cr}(\tilde{a} \star \tilde{b}) = \frac{1}{2}[\text{Pos}(\tilde{a} \star \tilde{b}) + \text{Nes}(\tilde{a} \star \tilde{b})]. \tag{5}$$

Lemma 1 *If $\tilde{a}$ and $\tilde{b}$ are two fuzzy numbers, then the credibility measure of the event $\tilde{a} < \tilde{b}$ is denoted by Cr($\tilde{a} < \tilde{b}$) and is given by [17]*

$$\text{Cr}(\tilde{a} < \tilde{b}) = \begin{cases} 1 & a_3 < b_1 \\ \frac{1}{2}\left(1 + \frac{b_2-a_2}{a_3-a_2+b_2-b_1}\right) & a_2 \le b_2 \quad \text{and} \quad b_1 < a_3 \\ \frac{1}{2}\left(\frac{b_3-a_1}{b_3-b_2+a_2-a_1}\right) & b2 < a2 \quad \text{and} \quad b_3 > a_1 \\ 0 & b_3 < a_1. \end{cases} \tag{6}$$

Fuzzy GTSP It is stated earlier that in a real-life GTSP, c_{ij} represents the cost of travel between city i and city j, or distance between city i and city j, or time to travel from city i to city j, etc. Though the distance between two cities is normally fixed, the time to travel from a city to another is always imprecise in nature in the fuzzy sense. The same property holds for traveling cost from a city to another. The distance of travel from a city to another also sometimes varies due to bad roadways or different/alternative ways. So in the general problem, it is better to consider c_{ij} as fuzzy number $\tilde{c}_{ij}$. In this research work, $\tilde{c}_{ij}$ is considered as a TFN $(c_{ij1}, c_{ij2}, c_{ij3})$. GTSP with imprecise $\tilde{c}_{ij}$ is named fuzzy GTSP.

3 Swap sequence-based particle swarm optimization for GTSP

PSOs are exhaustive search algorithms based on the emergent motion of a flock of birds searching for food [6,16] and has been extensively used/modified to solve complex decision-making problems in different fields of science and technology [1,7,9,11,12,28]. A PSO normally starts with a set of potential solutions (called swarm) of the decision-making problem under consideration. Individual solutions are called particles and location of food is the optimal solution. In simple terms, the particles are flown through a multi-dimensional search space, where the position of each particle is adjusted according to its own experience and that of its neighbors. Each particle i has a position vector $(X_i(t))$, a velocity vector $(V_i(t))$, the position at which the best fitness $(X_{\text{pbesti}}(t))$ has been encountered by the particle so far, and the best position of all particles $(X_{\text{gbest}}(t))$ in the current generation t. In generation $(t+1)$, the position and velocity of the particle are changed to $X_i(t+1)$ and $V_i(t+1)$ using the following rules:

$$V_i(t+1) = wV_i(t) + c_1r_1(X_{\text{pbesti}}(t) - X_i(t)) + c_2r_2(X_{\text{gbest}}(t) - X_i(t)), \quad (7)$$

$$X_i(t+1) = X_i(t) + V_i(t+1). \quad (8)$$

The parameters c_1 and c_2 are set to constant values, which are normally taken as 2, r_1 and r_2 are two random values uniformly distributed over [0, 1], and $w(0 < w < 1)$ is the inertia weight which controls the influence of previous velocity on the new velocity. It is mainly used to solve continuous optimization problems. It is also used to solve traveling salesman problems, where swap sequence and swap operations are used to find the velocity of a particle and its updating [20,35,38]. A PSO that uses swap sequence and swap operation is called SSPSO. As discussed in §2, in a GTSP a potential solution is represented by a sequence of cities and one and only one city is selected from a cluster. SSPSO is used to find the sequence of groups from which the cities are to be selected for a solution. In SSPSO, swap operations on different groups are used to update the sequence of groups of a solution. A swap sequence represents a sequence of swap operations used to transform the sequence of groups from a solution to another solution. The basic operations of SSPSO are briefly presented below:

Swap operator Consider a normal solution sequence of group in GTSP with n cities, and m groups $X = (x_1, x_2, \ldots, x_m)$, where $x_i \in \{g_1, g_2, \ldots, g_m\}$ and each g_i are distinct. Here, swap operator, $SO(i, j)$ is defined as exchange of group x_i and x_j in solution sequence X. Then we define $X' = X + SO(i, j)$ as a new sequence on the operating operator $SO(i, j)$ on X. So, the plus sign '+' above has its new meaning. We explain with a concrete example: suppose there is a GTSP problem with six groups, and $X = (x_1, x_2, x_3, x_4, x_5, x_6) = (g_1, g_3, g_5, g_2, g_4, g_6)$ be a sequence. Let the swap operator be $SO(2, 4)$, then $X' = X + SO(2, 4) = (g_1, g_3, g_5, g_2, g_4, g_6) + SO(2, 4) = (g_1, g_2, g_5, g_3, g_4, g_6)$, i.e. , groups of position 2 and position 4 are exchanged.

Swap sequence A swap sequence SS is made up of one or more swap operators. Let $SS = (SO_1, SO_2, \ldots, SO_p)$, where $SO_1, SO_2, \ldots, SO_p$ are swap operators. Swap sequence acting on a sequence of groups of a solution means all the swap operators of the swap sequence act on the sequence of groups in order. This can be described by the following formula:

$$X' = X + SS = X + (SO_1, SO_2, \ldots, SO_p) = (((X + SO_1) + SO_2) \cdots + SO_p).$$

Different swap sequences acting on the same sequence of groups may produce the same new sequence. All these swap sequences are named an equivalent set of swap sequences. In the equivalent set, the sequence which has the least swap operator is called basic swap sequence of the set or basic swap sequence (BSS) in short.

Several swap sequences can be merged into a new swap sequence. Here, the operator $\oplus$ is defined as merging two swap sequences into a new swap sequence. Suppose there are two swap sequences, $SS1$ and $SS2$, which act on one sequence X in order, namely $SS1$ first and $SS2$ second, a new sequence of groups X' is obtained. Let there be another swap sequence SS' acting on the same sequence of groups X and gets the sequence of groups X', then SS' is called merging of $SS1$ and $SS2$ and is represented as:

$$SS' = SS1 \oplus SS2.$$

Here, SS' and $SS1 \oplus SS2$ are in the same equivalent set.

The construction of basic swap sequence Suppose there are two solutions, A and B, and our task is to construct a basic swap sequence SS which can act on B to get solution A. Here, SS is defined as $SS = A - B$ (the sign $-$ also has its new meaning). One can swap the cities in B according to A from left to right to get SS. So, there must be an equation $A = B + SS$. For example, consider two solutions:

$$A = (g_1, g_2, g_3, g_4, g_5), \quad B = (g_2, g_3, g_1, g_5, g_4).$$

Here, $A(1) = B(3) = g_1$ and so the first swap operator is $SO(1, 3)$, $B1 = B + SO(1, 3)$; then we get the following result:

$$B1 : (g_1, g_3, g_2, g_5, g_4).$$

Again, $A(2) = B1(3) = g_2$; so the second operator is $SO(2, 3)$ and $B2 = (g_1, g_2, g_3, g_5, g_4)$. The third opera-

tor is $SO(4,5)$, then B3=A. Finally, we get the basic swap sequence $SS = A - B = (SO(1,3), SO(2,3), SO(4,5))$.

The transformation of the particle updating formulas For solving the GTSP formula, (7) and (8) of PSO have to be transformed using swap sequences and swap operations as follows:

$$V_i(t+1) = V_i(t) \oplus r_1 \odot (X_{\text{pbesti}}(t) - X_i(t)), \\ \oplus r_2 \odot (X_{\text{gbest}}(t) - X_i(t)) \quad (9)$$

$$X_i(t+1) = X_i(t) \oplus V_i(t+1). \quad (10)$$

Here, r_1, r_2 are random numbers between 0 and 1. Velocity $V_i(t)$ represents a swap sequence. $r_1 \odot (X_{\text{pbesti}}(t) - X_i(t))$ means all swap operators in BSS $(X_{\text{pbesti}}(t) - X_i(t))$ should be maintained with the probability of r_1, i.e., each swap operator in BSS $(X_{\text{pbesti}}(t) - X_i(t))$ should be selected with probability r_1. The same meaning is for the expression $r_2 \odot (X_{\text{gbest}}(t) - X_i(t))$. From here, it is seen that the bigger the value of r_1, the greater is the influence of $X_{\text{pbesti}}(t)$ for more swap operators.

4 Genetic algorithm for GTSP

GA are exhaustive search algorithms based on the mechanics of natural selection and genesis (crossover, mutation, etc.) and have been introduced by Holland [15] inspired by Darwin's theory in the 1970s. The idea behind GA is to model the natural evolution by using genetic inheritance together with Darwin's theory. Basic operations of GA are selection, crossover and mutation. In natural genesis, it is known that chromosomes are the main carriers of hereditary information from parent to offspring and that genes, which present hereditary factors, are lined up on chromosomes. At the time of reproduction, crossover and mutation take place among the chromosomes of parents. In this way, hereditary factors of parents are mixed up and carried to their offsprings. Again, Darwinian principle states that only the fittest animals can survive in nature. So, a pair of fittest parent normally reproduces a better offspring. The same phenomenon is followed to create a genetic algorithm for an optimization problem. A GA normally starts with a set of potential solutions (called initial population) of the decision-making problem under consideration. Individual solutions are called chromosomes and the fitness of each chromosome is evaluated by the evaluation process. Solutions are selected from the population randomly depending on their fitness by the selection process for the mating pool. Crossover and mutation operations happen among these selected solutions to get a new set of solutions and it continues until terminating conditions are encountered.

It has already been stated that to solve GTSP, PSO is used to determine the sequence of groups in which a salesman should travel. GA is used to find appropriate cities from different groups in a sequence of groups determined by PSO, so that the total cost spent by the salesman is a minimum for that particular sequence. Consider a GTSP with n cities $v_1, v_2, \ldots, v_n$ and m groups $g_1, g_2, \ldots, g_m$. Let $X = (x_1, x_2, \ldots, x_m)$, where $x_i \in \{g_1, g_2, \ldots, g_m\}$ be a sequence of groups for a solution which is determined by PSO in a particular iteration. The major steps for the selection of cities from different groups by GA for such a sequence are discussed below.

Initialization Initially, a set of N-solutions $I_{\text{pop}} = \{Y_1, Y_2, \ldots, Y_N\}$ is randomly generated. A solution is represented by an m-dimensional integer vector $Y_i = (y_{i1}, y_{i2}, \ldots, y_{im})$, where $y_{ij} \in x_j$ for $j = 1, 2, \ldots, m$ and are randomly selected from x_j.

Evaluation The fitness of a solution Y_i is taken as the ratio of the number of solutions of the population dominated by Y_i and the population size N and is represented by $f(Y_i)$. Let the total travel cost spent by a salesman for a solution Y_i be $C(Y_i)$ and for another solution Y_j be $C(Y_j)$. Then, $C(Y_i) = c_{y_{i1}y_{i2}} + c_{y_{i2}y_{i3}} + \cdots + c_{y_{im-1}y_{im}} + c_{y_{im}y_{i1}}$ and $C(Y_j) = c_{y_{j1}y_{j2}} + c_{y_{j2}y_{j3}} + \cdots + c_{y_{jm-1}y_{jm}} + c_{y_{jm}y_{j1}}$. Clearly for crisp GTSP, Y_i dominates Y_j if $C(Y_i) < C(Y_j)$. For fuzzy GTSP, with TFN type $\tilde{c}_{ij} = (c_{ij1}, c_{ij2}, c_{ij3})$, we have $\tilde{C}(Y_i) = (CY_{i1}, CY_{i2}, CY_{i3})$, a TFN, where $CY_{ik} = c_{y_{i1}y_{i2}k} + c_{y_{i2}y_{i3}k} + \cdots + c_{y_{im-1}y_{imk}} + c_{y_{im}y_{i1}k}$, $k = 1, 2, 3$. Here, Y_i dominates Y_j if $cr(\tilde{C}(Y_i) < \tilde{C}(Y_j)) > 0.5$ (cf. Lemma1). It is a valid fuzzy comparison, since $cr(\tilde{A} < \tilde{B}) + cr(\tilde{A} \geq \tilde{B}) = 1$. The comparison of objective values for the particles of PSO is made in the same manner.

Selection All solutions of the population do not take place in the evolution process (crossover and mutation). Mainly, solutions with higher fitness are selected for the process. Various types of selection techniques are available to select solutions from the initial population for the matting pool [10,23]. Here, roulette wheel (RW) selection technique [23]is used to select N solutions from the population for the mating pool depending on their fitness. The following are the steps of this selection:

(i) *Find the total fitness of the population* $F = \sum_{i=1}^{N} f(Y_i)$.
(ii) *Calculate the probability of selection* p_i *of each solution* Y_i *by the formula* $p_i = \frac{f(Y_i)}{F}$.
(iii) *Calculate the cumulative probability* cp_i *for each solution* Y_i *by the formula* $cp_i = \sum_{j=1}^{i} p_j$.
(iv) *Generate a random number r in the interval* (0, 1).
(v) *If* $r < cp_1$, *then select* Y_1; *otherwise select* $Y_i (2 \leq i \leq N)$ *where* $cp_{i-1} \leq r < cp_i$.
(vi) *Repeat steps (iv) and (v) N times to select N solutions for the mating pool. One solution may be selected more than once.*

(vii) *The selected solution set is denoted by* $N_{\text{pop}} = \{NY_1, NY_2, \ldots, NY_N\}$ *in the proposed algorithm.*

Crossover The crossover process involved the following steps:

(i) *Selection for crossover* For each solution of N_{pop} generate a random number r in the interval $(0, 1)$. If $r < pc$, then the solution is taken for crossover, where pc is the probability of crossover. The crossover operation takes place on every pair of such selected solutions.

(ii) *Selection for the crossover process* For every pair of selected solutions, one crossover operation is randomly selected from a set of two crossover operations-{single-point crossover and two-point crossover} for the crossover operation. Selection of this crossover process (i.e., either single point or two point) is made by a random process. The process is that: generate a random number r from the interval $(0, 1)$. if $r < 0.5$, then select a single-point crossover; otherwise, select a two-point crossover.

(iii) *Crossover process* Two crossover processes are discussed below:
Single-point crossover: For each pair of parent solutions, $PY_1 = (py_{11}, py_{12}, \ldots, py_{1m})$ and $PY_2 = (py_{21}, py_{22}, \ldots, py_{2m})$, selected for crossover, an integer position k is selected randomly in the range $[1, m]$, for crossover operation, where m is the length of a solution. Then two new solutions (offspring), CY_1, CY_2, created by a single-point crossover process on the parents, are given below [10] (c.f., Fig. 3):

$$CY_1 = (py_{11}, py_{12}, \ldots, py_{1k}, py_{2k+1} \ldots, py_{2m}),$$
$$CY_2 = (py_{21}, py_{22}, \ldots, py_{2k}, py_{1k+1} \ldots, py_{1m}).$$

Two-point crossover It is a similar process to single-point crossover, except that two cut points(positions) are randomly selected in the range $[1, m]$ for the crossover operation. Let the two cut points be k_1, k_2, where $1 < k_1 < k_2 < m$. Then, two new solutions (offsprings)

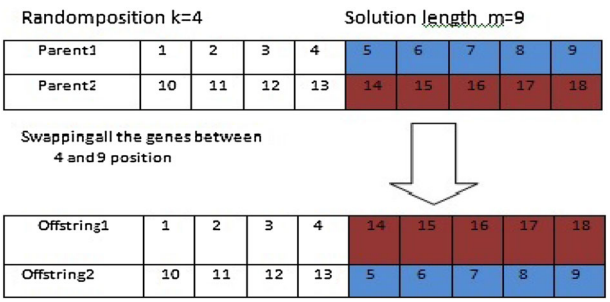

Fig. 3 Single-point crossover operation example

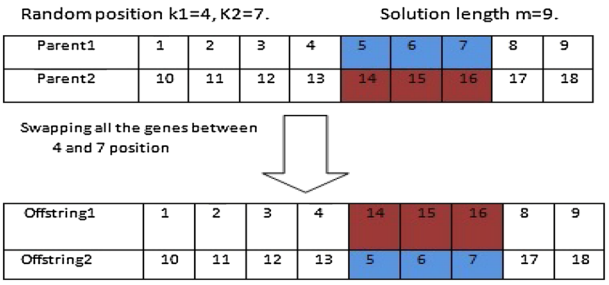

Fig. 4 Two-point crossover operation example

CY_1, CY_2, created by two-point crossover process, on the parents are given below [10](c.f., Fig. 4):

$$CY_1Z = (py_{11}, py_{12}, \ldots, py_{1k_1}, py_{2k_1+1}, py_{2k_1+2}, \ldots, py_{2k_2}, py_{1k_2+1}, py_{1k_2+1}, \ldots, py_{1m}),$$
$$CY_2 = (py_{21}, py_{22}, \ldots, py_{2k_1}, py_{1k_1+1}, py_{1k_1+2}, \ldots, py_{1k_2}, py_{2k_2+1}, py_{2k_2+1}, \ldots, py_{2m}).$$

(iv) *Upgradation of parents* If the path cost of CY_1 is smaller than that of PY_1, then replace PY_1 with CY_1. Similarly, if the path cost of CY_2 is smaller than that of PY_2, then replace PY_2 with CY_2.

Mutation The mutation process takes place on some selected solutions from the mating pool. Selection takes place according to the probability of mutation. The steps for this purpose are presented below:

(i) *Selection for mutation* For each solution of N_{pop}, generate a random number r from the interval $(0, 1)$. If $r < pm$, then the solution is taken for mutation, where pm is the probability of mutation.

(ii) *Mutation process* To mutate a selected solution $PY = (py_1, py_2, \ldots, py_m)$, select a random integer number k in the range $[1, m]$. Then py_k is replaced by a randomly selected city from its group to get a new mutated solution. This process is repeated for a finite number of iterations or until an improvement of the selected solution is made.

5 K-Opt operation for GTSP

K-Opt is a local search algorithm [2] mainly used for TSP which is based on the exchange of K parts (sub-tours) and their reverses (reverse sub-tours) of a tour (path) of the TSP under consideration to find a better tour. In the PSO part of the algorithm, it is tacitly used to find the optimal sequence of groups in which a salesman should select a particular city from a group in his tour to minimize the cost. If it is assumed

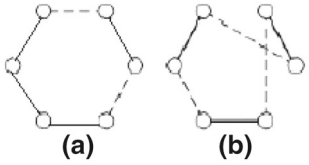

Fig. 5 All combinations of sub_toures for $k = 2$

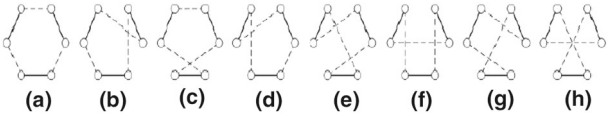

Fig. 6 All combinations of sub_toures for $k = 3$

that groups are connected by edges, then, while breaking (removes) K edges of a sequence of groups in a tour, there are $(K-1)!2^{K-1}$ ways to reconnect it (including the initial sequence) to form a valid sequence. Each new combination gives a new sequence of groups. We find the corresponding optimal tour using GA. Among these sequences, one may produce a better tour than the tour produced by the original sequence and can be taken as an improvement. In the case of the 2-Opt algorithm, we remove two edges from the sequence and reconnect all combinations of sub-sequences and their reverses (Fig. 5). Continue this process until no 2-Opt improvements can be found. Similarly in the case of 3-Opt, breaking three edges in a sequence there are in total eight cases of reconnection (Fig. 6). If a sequence is 3-optimal, it is also 2-optimal [2]. Continue break (remove) edges from the sequence, i.e., $K = 1, 2, 3, \ldots, n$ and get new algorithms, such as 2-Opt, 3-Opt, 4-Opt and so on. But increase of K increases the time complexity. Due to this, the 3-Opt operation is used and it is found that it acts better than the 2-Opt operation for large size GTSPs. In the proposed method, the 3-Opt operation is periodically used in a predefined number of iterations to improve the quality of the solutions in the PSO part of the algorithm and at the end of the algorithm.

6 Proposed algorithm for GTSP

A Hybrid algorithm based on SSPSO and GA algorithm with a local search (K-Opt) for GTSP Consider a GTSP with n cities $\{v_1, v_2, \ldots, v_n\}$ and m groups $\{g_1, g_2, \ldots, g_m\}$. The group g_k contains n_k cities, i.e., $\sum_{k=1}^{m} n_k = n$. In the algorithm, the i-th solution is represented by a triplet $\{X_i, Y_i, C(Y_i)\}$, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$, represents the sequence of groups in which travel is to be made, i.e., $x_{ij} \in \{g_1, g_2, \ldots, g_m\}$ and each x_{ij} are distinct. $Y_i = (y_{i1}, y_{i2}, \ldots, y_{im})$ represents the actual path corresponding to X_i, i.e., $y_{ij} \in x_{ij}$ for $j = 1, 2, \ldots, m$. $C(Y_i)$ is the path cost of Y_i. PSO is used to find the optimal sequence (X_i) of groups of a solution and GA is used to select cities from different groups to find the optimal path Y_i for X_i. Here, SS is the swarm size, $MaxGen$ is the positive integer that represents the maximum number of iterations for PSO and t is the iteration counter.

1. **Start Algorithm**
2. Set values of n, m, SS, $MaxGen$, and set $t = 0$.
3. **for** $k = 1$ to SS **do**
   Randomly generate $X_k(t)$
   Find $Y_k(t)$ using **GA algorithm** (§6.1)
   Determine $D(Y_k(t))$.
   end for
4. **for** $k = 1$ to SS **do**
   $X_{pbestk}(t) = X_k(t)$.
   $Y_{pbestk}(t) = Y_k(t)$.
   $V_k(t)$=SO(i, j) where i, j are randomly generated in range[1, m], and $i \neq j$.
   end for
5. X_{gbest}= $X_{pbestk}(t)$, where $D(Y_{pbestk}(t))$= $\min_{i\in\{1,2,..,SS\}} D(Y_{pbesti}(t))$.
6. $Y_{gbest}(t) = Y_{pbestk}(t)$.
7. **repeat**
   $t = t + 1$.
   for $k = 1$ to SS *do*
   Determine $V_k(t)$ using equation (9).
   Determine $X_k(t)$ using equation(10).
   Find $Y_k(t)$ using **GA algorithm** (§6.1)
   Determine $D(Y_k(t))$.
   If $mod(t, 10)$= 0
   Applying **K-Opt operation** on $X_k(t)$ along with GA for a predefined number iterations to improve the quality of final solutions (§6.2).
   end if
   end for
   for $k = 1$ to SS, *do*
   If $D(Y_{pbestk}(t-1)) > D(Y_k(t))$
   $X_{pbestk}(t) = X_k(t)$.
   $Y_{pbestk}(t) = Y_k(t)$.
   else
   $X_{pbestk}(t) = X_{pbestk}(t-1)$
   $Y_{pbestk}(t) = Y_{pbestk}(t-1)$
   end if
   If $D(Y_{gbest}) > D(Y_k(t))$
   $Y_{gbest} = Y_k(t)$.
   $X_{gbest} = X_k(t)$.
   end if
   end for
   Until ($t >$ $MaxGen$)
8. Applying **K-Opt operation** on X_{gbest} along with GA ina predefined number of iterations to improve its quality if possible.
9. **Output** X_{gbest}, Y_{gbest}, $D(Y_{gbest})$
10. **End of Algorithm**

6.1 GA algorithm for selection of cities from different groups to find the optimal path corresponding to a sequence of groups $X_k(t)$

In the algorithm, c is the iteration counter, Maxgen is the maximum number of iterations, pc is the probability of crossover, and pm is the probability of mutation.

i. Start Algorithm
ii. Set $pc = .8$, $pm = .6$, Maxgen=100 and c=0.
iii. Generate $I_{pop} = \{Y_1, Y_2, ..., Y_N\}$.
iv. Evaluate path length and fitness of each solution.
v. **Repeat**
   a. Select N solutions from I_{pop}, for mating pool using Roulette-wheel selection process (One solution may be selected more than once). Let this set be N_{pop}.
   b. Select solutions from N_{pop}, for crossover and mutation depending on pc and pm respectively.
   c. Make crossover on selected solutions for crossover.
   d. Make mutation on selected solutions for mutation.
   e. Set $I_{pop} = N_{pop}$.
   f. $c = c + 1$.
   Until($c > maxgen$).
vi. Return the path with smallest tour cost.
vii. End Algorithm

6.2 K-Opt operation on sequence of groups for improvement of sequence of groups $X_k(t)$

The detailed algorithm of K-Opt operation for $K = 3$ is presented below. In the algorithm, a one-dimensional array of size m, $X_{tem_k}(t)$, is used to represent a temporary sequence of group corresponding to $X_k(t)$ in iteration t. $Y_{tem_k}(t)$ is used to represent the actual path corresponding to $X_{tem_k}(t)$. $X_{ki}(t)$ and $X^r_{ki}(t), i = 1, 2, 3$ are one-dimensional arrays used to represent the sub-sequence of group and revers_sub-sequence of group of the original sequence of groups $X_k(t)$. $Maxit3$ is the maximum number of iterations.

for i = 1 to $Maxit3$ **do**

- Remove three edges (randomly selected) from the sequence of group $X_k(t)$, it makes three sub-sequences of group $X_{ki}(t), i = 1, 2, 3$.
- Reverses of the contents of these sub-sequence of groups are called revers_sub-sequence, represented as $X^r_{ki}(t)$, $i = 1, 2, 3$, i.e., $X^r_{k1}(t)$ = revers_sub-sequence of group$X_{k1}(t)$, $X^r_{k2}(t)$ = revers_sub-sequence of $X_{k2}(t)$, $X^r_{k3}(t)$ = revers_sub-sequence of $X_{k3}(t)$.
- Now, combining the sub-sequences of groups $\{X_{k1}(t), X_{k2}(t), X_{k3}(t)\}$, $\{X^r_{k1}(t), X^r_{k2}(t), X^r_{k3}(t)\}$, a new sequence of groups can be formed in the following eight combinations:

i. $\{X_{k1}(t), X_{k2}(t), X_{k3}(t)\}$
ii. $\{X_{k1}(t), X^r_{k2}(t), `X_{k3}(t)\}$
iii $\{X_{k1}(t), X_{k2}(t), X^r_{k3}(t)\}$
iv $\{X_{k1}(t), X^r_{k3}(t), X^r_{k2}(t)\}$
v $\{X_{k1}(t), X_{k3}(t), X^r_{k2}(t)\}$
vi $\{X_{k1}(t), X^r_{k3}(t), X_{k2}(t)\}$
vii $\{X_{k1}(t), X^r_{k2}(t), X^r_{k3}(t)\}$
viii $\{X_{k1}(t), X^r_{k3}(t), X_{k2}(t)\}$

for each combination do
  Create a new sequence of group from the combination and let it be $X_{tem_k}(t)$.
  Find $Y_{tem_k}(t)$ using **GA algorithm** (§6.1).
  Determine $C(Y_{tem_k}(t))$.
  if $C(Y_{tem_k}(t)) < C(Y_k(t))$
    $X_k(t) = X_{tem_k}(t)$
    $Y_k(t) = Y_{tem_k}(t)$
  end if
end for

7 Experimental results

All computational experiments are conducted with Dev C++ 5.8.3, core $i3$ CPU @ 2.10 GH_z, Windows 8.1 Operating System and 4 GB RAM. The performance of the proposed algorithm is tested using different size standard GTSPs from TSPLIB. Each problem algorithm is tested by running the program five times for different seeds of random number generator and the best solution is selected. The average cost of these solutions and percentage of relative error (Error(%)) according to the cost of optimal solution are calculated. The percentage of relative **Error(%)** is calculated using the following equation.

$$\textbf{Error}\ (\%) = \frac{\text{average cost of solutions} - \text{cost of the optimal solution}}{\text{cost of the optimal solution}} \times 100. \quad (11)$$

The results obtained by the proposed algorithm for 21 different test problems from TSPLIB are presented in Table 1 (Fig. 7).

In Table 1, the 1st column stands for the problem name, 2nd (n) for the number of cities, 3rd (m) for the number

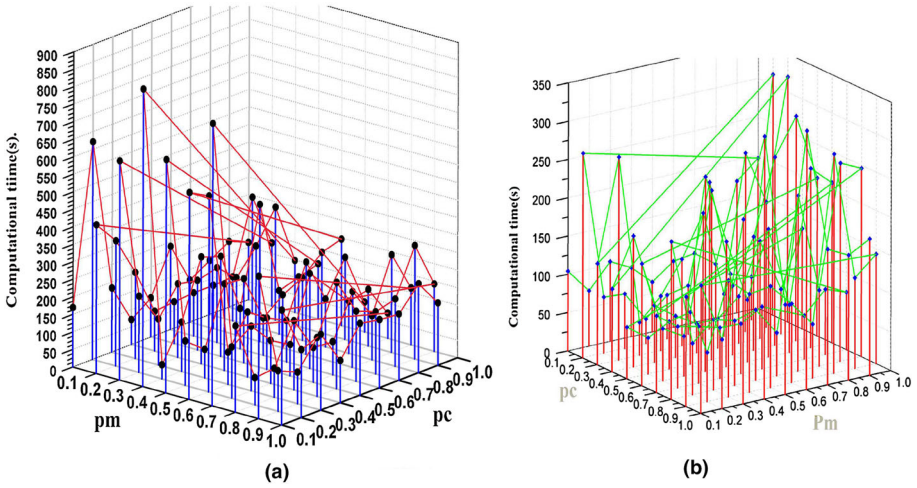

Fig. 7 Computational time with respect to pc and pm for problems 29PR144(**a**) and 25PR124(**b**)

of groups in the GTSP, 4th for the cost of optimal solution for each problem [29], 5th for the cost of the best solution obtained by the proposed method, 6th–10th for the obtained results in different runs of the algorithm using different seeds of random number generator for each problem, 11th and 12th columns provide some statistical information about the problem, such as average cost of the obtained solutions and percentage of relative error of the average cost of the solutions. The last column presents the average computational time in seconds. It is found from Table 1 that the solution produced by the algorithm for each of the considered problems is the same as its optimal solutions [29]. As a result, the obtained average cost of solutions of each problem is the same as the cost of the optimal solution of the corresponding problem. The relative error of each problem is exactly zero, i.e., the proposed algorithm obtained exact solution for each problem considered for testing. As GTSP belongs to the NP-hard problem group, it is not possible to find the time complexity of the algorithm used for finding its solution. But a graph is drawn using different size problems and corresponding computational time required for finding the optimal solution (cf. Fig. 8). As expected, it is clear from Fig. 8 that computational time increases with problem size. It should be noted that computational time depends on several factors like processor speed, operating system, size of catch memory, different software running (in back-ground) in the machine, etc.

Again, solutions (paths and corresponding costs) obtained in different runs of the algorithm using different seeds of random number generator are analyzed and it is observed that for many problems there are multiple optimal paths. The different optimal paths for some problems are listed in Table 2.

Fuzzy GTSP For fuzzy GTSP, no standard test problems are available in the literature. So, here, crisp GTSP problems from TSPLIB are used to generate fuzzy GTSP problems. Only TFNs are used to represent costs of a fuzzy GTSP. Let c_{ij} be the cost of travel from city i to city j for a crisp GTSP from TSPLIB. For the corresponding fuzzy GTSP, fuzzy cost of travel $\tilde{c}_{ij}$ from city i to city j is considered as $\tilde{c}_{ij} = (c_{ij1}, c_{ij2}, c_{ij3})$, where $c_{ij2} = c_{ij}$, $c_{ij1} = c_{ij} - R1$, $c_{ij3} = d_{ij} + R2$, and $R1$ and $R2$ are randomly generated in the interval $(0, R \times c_{ij}/100)$. For 5% fuzziness R is taken as 5, for 10% fuzziness R is taken as 10, etc. From each crisp GTSP, three different fuzzy GTSPs are generated considering $R = 5$, $R = 10$ and $R = 15$, respectively. After generating a problem, it is solved using the proposed algorithm for fuzzy GTSP and the results obtained are presented in Table 3. It is found from Table 3 that for small size problems and for small amount of fuzziness (5%), a mid value of optimum cost as well as optimal path is same as the corresponding crisp GTSP. While fuzziness increases, the optimum cost as well as optimum path varies gradually. All these observations agree with reality. The last column of Table 3 presents the computational time (s) for the problems with 5 % fuzziness.

Table 1 Computational results obtained by different runs of the algorithm due to considered test problems from TSPLIB

Problem from TSPLIB	Number of cities (n)	Number of groups (m)	Optimal cost of the problem	Best cost obtained	Cost of the solution obtained in different runs					Average cost of the solutions	Error (%)	Computational time in second
					Run1 seed =1	Run2 seed = 2	Run3 seed = 3	Run4 seed = 4	Run5 seed = 5			
10ATT48	48	10	5394	5394	5394	5394	5394	5394	5394	5394.00	0.00	2.15
11EIL51	51	11	174	174	174	174	174	174	174	174.00	0.00	2.52
14ST70	70	14	316	316	316	316	316	316	316	316.00	0.00	3.42
16EIL76	76	16	209	209	209	209	209	209	209	209.00	0.00	5.62
16PR76	76	16	64,925	64,925	64,925	64,925	64,925	64,925	64,925	64,925.00	0.00	6.27
20RAT99	99	20	497	497	497	497	497	497	497	497.00	0.00	18.27
20KROA100	100	29	9711	9711	9711	9711	9711	9711	9711	9711.00	0.00	25.42
20KROB100	100	20	10,328	10,328	10,328	10,328	10,328	10,328	10,328	10,328.00	0.00	20.48
20KROC100	100	20	9554	9554	9554	9554	9554	9554	9554	9554.00	0.00	22.30
20KROD100	100	20	9450	9450	9450	9450	9450	9450	9450	9450.00	0.00	27.64
20KROE100	100	20	9523	9523	9523	9523	9523	9523	9523	9523.00	0.00	24.25
21EIL101	101	21	249	249	249	249	249	249	249	249.00	0.00	38.64
22PR107	107	22	27,898	27,898	27,898	27,898	27,898	27,898	27,898	27,898.00	0.00	33.96
25PR124	124	24	36,605	36,605	36,605	36,605	36,605	36,605	36,605	36,605.00	0.00	56.63
26BIER127	127	26	72,418	72,418	72,418	72,418	72,418	72,418	72,418	72,418.00	0.00	68.25
28PR136	136	28	42,570	42,570	42,570	42,570	42,570	42,570	42,570	42,570.00	0.00	74.28
29PR144	144	29	45,886	45,886	45,886	45,886	45,886	45,886	45,886	45,886.00	0.00	94.32
30KROB150	150	30	11,018	11,018	11,018	11,018	11,018	11,018	11,018	11,018.00	0.00	98.65
31PR152	152	31	51,576	51,576	51,576	51,576	51,576	51,576	51,576	51,576.00	0.00	150.32
40D198	198	40	10,557	10,557	10,557	10,557	10,557	10,557	10,557	10,557.00	0.00	198.62
46PR226	226	46	64,007	64,007	64,007	64,007	64,007	64,007	64,007	64,007.00	0.00	205.52

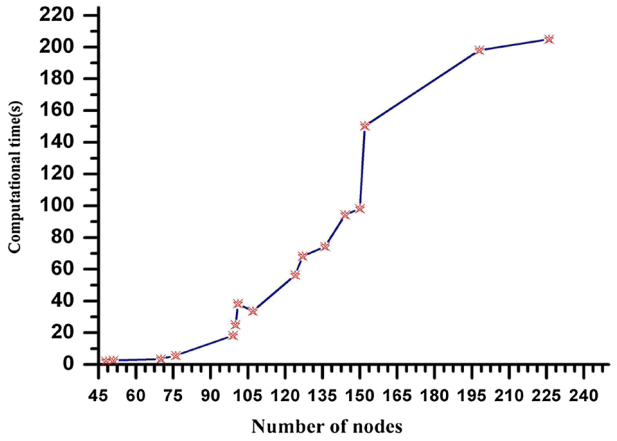

Fig. 8 Problem size versus computational time

For fuzzy GTSP, also it is found that for some problems there are multiple optimal paths. For multiple paths, the mid values of the fuzzy objective (TFN) for all the optimal paths of a problem are the same. The other two components (left and right limits) differ by a very small amount (cf. Table 4).

Table 5 represents a comparison of some computational results of the proposed algorithm with other existing algorithms in the literature. In Table 5, the results of PSO [32], GCGA [36] and GA [34] are taken from the corresponding reference paper and the results of the proposed method are taken from a single run of the algorithm for the corresponding problem. From Table 5, it is clear that the proposed approach is better compared to some other existing approaches in the literature with respect to the accuracy (Error(%)). For the test problems like 14ST70,

Table 2 Different optimal paths of some test problems from TSPLIB obtained by the algorithm

Problem form TSPLIB	Different optimal path	Cost of the solution
	45,41,25,24,27,1,22,20,50,10,33	
	27,24,25,41,44,33,10,50,20,22,1	
11EIL51	20,16,49,33,45,41,25,24,27,1,22	174
	44,33,9,16,20,22,1,27,24,25,41	
	41,45,33,49,29,20,22,1,27,24,25	
	23,69,59,57,26,8,44,43,61,39,54,11,64,16	
	44,43,61,39,48,11,64,16,23,69,59,19,26,8	
14ST70	11,64,16,23,69,59,57,4,7,26,44,61,39,54	316
	43,44,8,26,19,59,69,23,16,64,11,54,39,61	
	8,44,43,61,39,62,11,64,16,23,69,59,19,26	
	59,11,38,39,9,25,49,41,42,62,2,48,37,15,13,14	
	9,25,49,41,42,22,47,37,27,52,46,8,59,11,58,39	
16EIL76	9,39,38,11,59,8,52,45,29,37,47,22,42,41,49,25	209
	14,13,15,37,48,30,62,42,23,49,25,9,39,58,11,59	
	59,8,52,45,5,37,47,42,43,23,49,25,9,39,72,11	
	26,35,34,53,71,80,89,87,95,83,65,56,57,48,30,11,2,4,7,17	
	7,4,2,11,30,48,57,56,65,83,95,87,89,80,71,53,34,35,26,17	
20RAT99	11,30,48,57,56,65,83,95,87,89,80,62,53,34,35,26,17,7,4,2	497
	7,5,2,11,30,48,57,56,65,83,95,87,89,80,71,53,34,35,26,17	
	65,83,95,87,89,80,71,53,34,35,26,17,7,4,2,11,30,48,57,56	
	24,84,47,98,77,83,29,48,78,96,82,44,73,69,67,8,92,75,66,18	
	77,83,29,48,78,96,82,44,73,69,67,8,92,75,66,18,24,84,47,98	
20KROA100	96,78,48,29,83,77,98,47,84,24,18,66,75,92,8,67,69,73,44,82	9711
	92,75,66,18,24,84,47,98,77,83,29,48,78,96,82,44,73,69,67,8	
	2,75,66,18,24,84,47,98,77,83,29,48,78,96,82,44,73,69,67,8	
	49,64,11,10,30,20,35,78,29,54,39,23,41,15,14,44,85,6,60,8,47	
	39,23,41,57,14,44,93,6,83,8,47,49,64,11,10,30,20,35,78,29,54	
21EIL101	44,14,42,22,23,39,54,29,78,35,20,30,10,11,64,49,47,8,60,6,96	249
	11,64,49,47,8,60,6,85,44,14,42,22,23,39,54,29,34,35,20,30,10	
	30,10,11,64,49,47,8,60,6,85,44,14,57,41,23,39,54,29,34,35,20	

$20KROD100$, $22PR107$, $25PR124$, and $40D198$, other algorithms also produce optimal solution, but with respect to the statical information(error) the proposed algorithm is better compared to some other algorithms in the literature. On the other hand, GCGA [36] provide results in least computational time, but its success rate is not 100%. Compared with GA [34], the proposed algorithm does not always provide optimum results in lesser computational time. In some problems, our algorithm takes less computational time and in some other problems GA [34] takes less time. But it should be noted that computational time depends on several factors such as processor speed, operating system, size of catch memory, and different software running (in the background) in the machine. The PSO [32] and GCGA [36] produce optimal results in some problems, but they are not capable of producing optimal solution in all runs of the algorithms for different test problems (at least up to a moderate size). For this reason, in this paper PSO and GA are combined together to create a hybrid algorithm to improve the efficiency of the algorithm.

Probability of crossover (pc) and probability of mutation (pm) are two important parameters of GA, and the performance of GA on a particular problem mostly depends on them. Though these parameters are problem dependent, three test problems (relatively large sizes) are used to check the performance (with respect to computation time) of GA with respect to these parameters and a comparative study table is presented in Table 6. The result is obtained for a particular seed of random number generator. From the table, it is observed that the algorithm takes minimum run time to produce optimal solution of the test problems for $pc = 0.8$ and $pm = 0.6$. Due to this reason, these values of pc and pm are taken for obtaining the results of other problems also. A graphical representation of the requirement of time to produce optimal solution due to different pc and pm for the two test problems—25PR124 and 29PR144—are presented in Fig. 7.

Table 7 represents the results obtained by the proposed method for different test problems using 2-Opt and 3-Opt operations in the algorithm. It is observed that, for small size problems like $11EIL51$ and $14ST70$, both the approaches produce the same solution as the optimal one. For large size problems, the algorithm produces different solutions using 2-Opt and 3-Opt. Problems for which optimal solutions are obtained by the algorithms are presented in boldface in Table 7. It is clear from Table 7 that for all the problems, the algorithm with 3-Opt produces better result than that using 2-Opt. So in the proposed algorithm, 3-Opt operation is used.

Table 3 Results obtained for different fuzzy TSPs generated from some problems of TSPLIB by the proposed algorithm

Problems from TSPLIB	Number of cities (n)	Number of groups (m)	Optimal cost of the problem	Results obtained after introduction of different amount of fuzziness			Computational time in second
				5% fuzziness	10% fuzziness	15% fuzziness	
10ATT48	48	10	5394	(5364.68, 5394.00, 5428.78)	(5333.65, 5394.00, 5455.45)	(5301.65, 5394.00, 5489.45)	5.24
11EIL51	51	11	174	(143.62, 174.00, 202.78)	(113.24, 174.00, 231.57)	(82.86, 174.00, 260.36)	9.15
14ST70	70	14	316	(287.48, 316.00, 346.64)	(253.56, 316.00, 384.99)	(230.46, 316.00, 407.94)	52.62
16EIL76	76	16	209	(166.83, 209.00, 253.21)	(143.83, 209.00, 307.12)	(69.91, 209.00, 331.18)	67.97
20RAT99	99	20	497	(449.95, 499.00, 540.89)	(414.27, 497.00, 598.81)	(373.85, 497.00, 655.16)	105.25
20KROA100	100	20	9711	(9673.45, 9711.00, 9751.23)	(9643.87, 9713.00, 9794.45)	(9619.53, 9714.00, 9831.43)	110.32
21EIL101	101	21	249	(205.45, 249.00, 300.63)	(163.83, 250.00, 372.62)	(119.66, 252.00, 409.43)	99.65
22PR107	107	22	27,898	(27863, 27898, 27937)	(27829, 27900, 27982)	(27795, 27900, 28023)	125.84
26BIER127	127	26	72,418	(72373.53, 72422.00, 72457.76)	(72345.65, 72431.00, 72500.43)	(72309.34, 72439.00, 72549.34)	160.42
28PR136	136	28	42,570	(42547.39, 42579.00, 42624.67)	(42520.76, 42585.00, 42667.30)	(42493.49, 42588.00, 42697.38)	205.65
29PR144	144	29	45,886	(45852.67, 45893.56, 42928.56)	(458826.65, 45905.00, 45836.87)	(45792.54, 45902.00, 45987.25)	250.75

Table 4 Different paths obtained for different fuzzy TSPs generated from some problems of TSPLIB by the proposed algorithm

Problem from TSPLIB	Amount of fuzziness introduced (%)	Different optimal paths obtained	Cost of the solution
11EIL51	5	24, 27, 1, 22, 20, 29, 10, 33, 45, 41, 25	(143.62, 174.00, 202.78)
		44, 33, 9, 16, 20, 22, 1, 27, 24, 25, 41	(143.87, 174.00, 203.42)
		29, 49, 33, 44, 41, 25, 24, 27, 1, 22, 20	(143.83, 174.00, 204.05)
		24, 27, 1, 22, 20, 29, 10, 33, 45, 41, 25	(113.24, 174.00, 231.57)
	10	44, 33, 9, 16, 20, 22, 1, 27, 24, 25, 41	(113.74, 174.00, 232.84)
		29, 49, 33, 44, 41, 25, 24, 27, 1, 22, 20	(113.66, 174.00, 234.10)
		24, 27, 1, 22, 20, 29, 10, 33, 45, 41, 25	(82.86, 174.00, 260.36)
	15	44, 33, 9, 16, 20, 22, 1, 27, 24, 25, 41	(83.62, 174.00, 262.26)
		29, 49, 33, 44, 41, 25, 24, 27, 1, 22, 20	(83.50, 174.00, 264.15)
		16, 64, 11, 54, 39, 40, 68, 8, 26, 4, 57, 59, 69, 23	(287.48, 316.00, 346.64)
	5	16, 64, 11, 54, 39, 9, 68, 8, 26, 4, 57, 59, 69, 23	(290.87, 316.00, 348.68)
		9, 68, 8, 26, 4, 57, 59, 69, 23, 16, 64, 11, 54, 39	(290.87, 316.00, 348.68)
		16, 64, 11, 54, 39, 40, 68, 8, 26, 4, 57, 59, 69, 23	(253.56, 317.00, 384.99)
14ST70	10	23, 16, 64, 11, 54, 39, 61, 43, 44, 8, 26, 19, 59, 69	(247.75, 316.00, 378.79)
		26, 8, 68, 40, 39, 54, 11, 64, 16, 23, 69, 59, 57, 4	(243.65, 316.00, 370.76)
		16, 64, 11, 54, 39, 40, 68, 8, 26, 4, 57, 59, 69, 23	(230.46, 316.00, 407.94)
	15	16, 64, 11, 54, 39, 9, 68, 8, 26, 4, 57, 59, 69, 23	(240.63, 316.00, 414.05)
		57, 59, 69, 23, 16, 64, 11, 54, 39, 40, 68, 8, 26, 4	(230.46, 316.00, 407.94)
		58, 11, 59, 8, 52, 45, 5, 37, 28, 42, 41, 63, 16, 3, 25, 39	(166.90, 211.00, 253.17)
	5	22, 42, 23, 49, 50, 25, 39, 72, 11, 59, 8, 52, 45, 5, 37, 28	(166.83, 209.00, 253.21)
		49, 50, 25, 39, 58, 11, 59, 8, 46, 52, 27, 37, 47, 22, 42, 23	(161.09, 209.00, 251.97)
		72, 11, 59, 8, 52, 45, 29, 37, 21, 42, 41, 63, 16, 3, 25, 39	(130.31, 211.00, 300.85)
16EIL76	10	47, 37, 29, 45, 52, 8, 59, 11, 58, 39, 9, 25, 49, 41, 42, 22	(143.83, 209.00, 307.12)
		8, 59, 11, 72, 39, 25, 50, 49, 41, 42, 22, 28, 37, 29, 45, 52	(155.31, 209.00, 308.70)
		37, 47, 22, 42, 41, 49, 50, 25, 39, 58, 11, 59, 8, 46, 52, 27	(69.91, 209.00, 331.18)
	15	22, 42, 23, 49, 25, 9, 39, 58, 11, 59, 8, 52, 45, 5, 37, 28	(78.31, 209.00, 332.85)
		49, 50, 25, 39, 58, 11, 59, 8, 52, 45, 29, 37, 28, 42, 43, 23	(72.93, 209.00, 331.02)
		74, 56, 57, 39, 30, 11, 2, 4, 7, 17, 26, 34, 44, 53, 71, 80, 89, 87, 95, 83	(449.95, 499.00, 540.89)
	5	74, 56, 57, 39, 30, 11, 2, 4, 7, 17, 26, 34, 44, 53, 71, 80, 89, 87, 95, 83	(449.95, 499.00, 540.89)
		71, 53, 52, 43, 35, 26, 17, 7, 4, 2, 11, 30, 57, 56, 65, 83, 95, 97, 88, 80	(453.41, 502.00, 561.27)
		74, 56, 57, 39, 30, 11, 2, 4, 7, 17, 26, 34, 44, 53, 71, 80, 89, 87, 95, 83	(405.95, 499.00, 611.89)
20RAT99	10	52, 43, 35, 17, 7, 5, 2, 11, 22, 31, 57, 56, 65, 83, 95, 87, 89, 80, 71, 53	(411.44, 504.00, 612.10)
		48, 57, 56, 74, 83, 95, 87, 89, 80, 71, 53, 34, 35, 26, 17, 7, 4, 2, 11, 30	(414.27, 497.00, 598.81)
		62, 80, 88, 97, 95, 83, 74, 56, 57, 38, 20, 2, 4, 7, 17, 26, 35, 34, 52, 53	(333.85, 503.00, 658.96)
	15	2, 5, 7, 17, 26, 35, 34, 53, 71, 80, 79, 97, 95, 83, 65, 56, 57, 48, 30, 11	(363.16, 498.00, 635.97)
		71, 53, 34, 35, 26, 17, 7, 5, 2, 11, 30, 48, 57, 56, 65, 83, 95, 87, 89, 80	(373.85, 497.00, 655.16)

Table 5 Comparison of results obtained by the proposed algorithm with respect to some other existing algorithms in the literature

Problems from TSPLIB	Cost of optimal path	Cost of the best path and the corresponding									
		Error (%) obtained using									
		PSO	Error (%)	GA	Error (%)	Time (S.)	GCGA	Error (%)	Time (S.)	Proposed algorithm	Error (%)	Time (S.)
11EIL51	**174**	176	**3.74**	174	**0.00**	**0.87**	176	**1.15**	2.52	**174**	**0.00**	2.52
14ST70	**316**	315	**0.62**	**316**	**0.00**	**7.3**	**316**	**0.00**	1.23	**316**	**0.00**	3.42
16EIL76	**209**	214	**5.46**	**209**	**0.00**	**9.4**	214	**2.87**	1.45	**209**	**0.00**	5.62
20KROA100	**9711**	9758	**3.26**	**9711**	**0.00**	**18.4**	9758	**1.15**	1.93	**9711**	**0.00**	25.42
20KROD100	**9450**	**9450**	**2.46**	**9450**	**0.00**	**14.3**	**9450**	**0.00**	1.96	**9450**	**0.00**	27.64
21EIL101	**249**	254	**9.69**	**249**	**0.00**	**25.6**	251	**0.08**	1.79	**249**	**0.00**	38.64
22PR107	**27,898**	27,901	**1.56**	**27,898**	**0.00**	**7.4**	**27,898**	**0.04**	2.25	**27,898**	**0.00**	33.96
25PR124	**36,605**	36,782	**2.20**	**36,605**	**0.00**	**25.9**	**36,605**	**2.12**	2.51	**36,605**	**0.00**	56.63
29PR144	**45,886**	45,890	**4.04**	**45,886**	**0.00**	**8.2**	45,890	**0.34**	2.67	**45,886**	**0.00**	94.32
30KROB150	**11,018**	11,085	**7.93**	**11,018**	**0.00**	**60.6**	11,117	**3.53**	3.26	**11,018**	**0.00**	98.65
40D198	**10,557**	10,693	**5.20**	**10,557**	**0.00**	**763.1**	**10,557**	**0.42**	3.50	**10,557**	**0.00**	198.62

Better results are in bold

Table 6 Time required (in s) to find the optimal solution of different test problems from TSPLIB using different values of pm and pc

Problem from TSPLIB	Probability of mutation (pm)	Probability of crossover (pc)									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	**0.8**	0.9	1
25PR124		196.92	314.46	320.28	131.28	206.24	209.47	75.68	246.24	143.72	243.85
29PR144	1	163.42	180.58	148.25	57.95	246.65	85.46	72.25	162.24	192.72	126.35
30KROB150		186.52	242.65	140.35	141.46	259.12	143.15	138.62	293.35	320.52	190.88
25PR124		172.4	56.99	163.5	58.28	293.25	102.47	85.68	259.24	118.65	178.12
29PR144	0.9	187.32	332.98	322.12	141.32	230.65	234.32	161.32	278.32	323.32	265.35
30KROB150		181.65	223.52	134.52	144.65	255.68	135.95	221.12	428.32	308.52	190.88
25PR124		70.25	66.85	233.27	127.95	246.75	62.95	57.65	99.14	286.65	163.87
29PR144	0.8	154.42	126.58	232.25	123.95	179.65	136.46	286.25	126.24	192.72	254.35
30KROB150		116.25	157.65	123.54	251.94	240.32	167.65	168.32	210.65	391.32	205.98
25PR124		90.52	192.62	55.36	61.75	148.45	167.86	81.65	229.64	70.35	177.65
29PR144	0.7	119.32	123.45	163.65	391.14	151.65	263.51	148.95	216.45	177.65	189.40
30KROB150		225.65	259.42	145.65	194.35	129.64	165.40	192.32	189.65	291.38	192.98
25PR124		86.58	63.85	167.65	38.65	90.25	81.32	65.95	**56.63**	103.65	388.25
29PR144	**0.6**	237.32	123.54	213.65	115.65	106.32	118.45	270.98	**94.32**	200.45	194.32
30KROB150		134.65	264.65	140.35	162.65	242.65	116.65	201.25	**98.65**	115.65	225.45
25PR124		65.32	57.32	113.25	38.65	221.32	163.65	66.65	92.32	132.65	117.32
29PR144	0.5	111.62	119.12	198.45	285.65	204.65	147.85	89.62	99.32	303.32	205.46
30KROB150		125.32	210.65	152.65	120.95	164.65	152.65	91.35	110.68	215.55	320.20
25PR124		90.32	32.65	71.95	54.65	68.96	254.65	74.65	84.65	128.65	343.65
29PR144	0.4	71.65	113.64	210.46	305.87	246.65	185.46	122.94	197.41	167.72	120.10
30KROB150		145.32	162.65	192.54	230.98	105.66	243.65	119.35	148.65	240.65	192.65
25PR124		104.25	71.25	43.36	74.62	62.52	52.62	80.52	92.62	152.65	241.82
29PR144	0.3	182.45	245.65	131.65	141.62	81.74	91.75	184.46	82.65	90.12	216.85
30KROB150		256.65	305.65	190.45	165.98	225.85	298.65	115.85	245.10	192.45	323.21
25PR124		72.36	73.62	265.65	171.63	47.32	68.32	100.62	67.32	64.65	90.25
29PR144	0.2	621.45	357.25	217.65	40.65	126.98	559.65	146.65	77.62	155.95	194.65
30KROB150		512.15	242.65	165.25	141.46	259.12	143.15	162.35	299.35	312.52	250.88
25PR124		104.36	267.32	132.28	108.32	67.36	159.36	215.32	206.32	143.72	167.32
29PR144	0.1	169.25	424.35	625.65	848.64	665.32	590.32	805.52	246.32	406.32	328.20
30KROB150		632.32	325.65	140.35	163.35	259.12	142.35	112.75	393.35	220.52	170.88

Better results are in bold

Table 7 Performance of the algorithm using 2-Opt and 3-Opt in different test problems from TSPLIB

Problem from TSPLIB	Optimal cost of the solution	Cost of the best path obtained by the hybrid algorithm	
		PSO+ GA+ 2−Opt	PSO+GA+ 3−Opt
11EIL51	**174**	**174**	**174**
14ST70	**316**	**316**	**316**
16EIL76	**209**	212	**209**
20KROA100	**9711**	9775	**9711**
20KROD100	**9450**	9502	**9450**
21EIL101	**249**	255	**249**
22PR107	**27,898**	27,905	**27,898**
25PR124	**36,605**	36,792	**36,605**
29PR144	**45,886**	45,965	**4588**

8 Conclusion

For the first time combining the features of SSPSO technique and GA, a hybrid algorithm is developed to solve GTSP in different environments. SSPSO is used to find the sequence of groups in a solution and for each sequence cities from different groups are selected using GA. Here K-Opt algorithm (for $K = 3$) is used periodically in a predefined number of iterations in the PSO portion of the algorithm to improve the quality of the solutions. It is tacitly used to find the optimal sequence of groups in which a salesman should select a particular city from a group in his tour to minimize the tour cost. Existing algorithms for GTSP are not suitable for GTSP in an imprecise environment. The proposed algorithm is capable of solving the problem in crisp as well as in imprecise environment. The efficiency of the algorithm is tested in crisp environment using different size benchmark problems available in TSPLIB [29]. In crisp environment, the algorithm gives 100% success rate for problems up to considerably large problem sizes. As the results of fuzzy GTSP and crisp GTSP are the same for relatively small size problems with small amount of fuzziness, it can be concluded that the algorithm is efficient for solving fuzzy GTSP. The algorithm can be used to solve GTSPs in different other imprecise environments, like stochastic environment, rough environment, etc. The only difference is that a comparison criteria of rough objectives have to be developed using a valid comparison operator like trust measure [24]. In stochastic environment, random objectives have to be be compared using a valid comparison technique like the chance constraints approach [25].

References

1. Angeline, P.J.: Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. In: Evolutionary Programming VII: proceedings of the seventh annual conference on evolutionary programming
2. Blazinskas, A., Misevicius, A.: Combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem, Kaunas University of Technology, Department of Multimedia Engineering, Studentu St. 50-401, 416a, Kaunas, Lithuania
3. Bontouxa, B., Artigues, C., Feilletc, D.: A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesmanProblem. Comput. Oper. Res. **37**, 1844–1852 (2010)
4. Changdar, C., Maiti, M.K., Maiti, M.: A constrained solid TSP in fuzzy environment: two heuristic approaches. Iran. J. Fuzzy Syst. **10**(1), 1–28 (2013)
5. Changdar, C., Mahapatra, G.S., Pal, R.: An efficient genetic algorithm for multi-objective solid traveling salesman problem under fuzziness. Swarm Evol. Comput. **15**, 27–37 (2014)
6. Eberhart, R., Kennedy, J.: A new optimizer using particles swarm theory. In: Roc. sixth international symposium on micro machine and human science (Nagoya, Japan) IEEE Service Center, Piscataway, NJ, pp. 39–43 (1995)
7. Fan, H.: Discrete particle swarm optimization for TSP based on neighborhood. J. Comput. Inf. Syst. (JCIS) **6**, 3407–3414 (2010)
8. Fischetti, M., Salazar, J.J., Toth, P.: A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Oper. Res. **45**, 378–394 (1997)
9. Giri, P.K., Maiti, M.K., Maiti, M.: Profit maximization solid transportation problem under budget constraint using fuzzy measures. Iran. J. Fuzzy Syst. **13**(5), 35–63 (2016)
10. Goldberg, D.E.: Genetic Algorithms: Search, Optimization and Machine Learning. Addison Wesley, Massachusetts (1989)
11. Guchhait, P., Maiti, M.K., Maitia, M.: Two storage inventory model of a deteriorating item with variable demand under partial credit period. Appl. Soft Comput. **13**, 428–448 (2013)
12. Guchhait, P., Maiti, M.K., Maitia, M.: Inventory model of a deteriorating item with price and credit linked fuzzy demand: a fuzzy differential equation approach. OPSEARCH **51**(3), 321–353 (2013)
13. Gutin, G., Karapetyan, D.: A memetic algorithm for the generalized traveling salesman problem. Nat. Comput. **9**, 47–60 (2010)
14. Henry-Labordere, A.L.: The record balancing problem: a dynamic programming solution of a generalized traveling salesman problem. RAIRO Oper. Res. **B2**, 43–49 (1969)
15. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
16. Kennedy, J., Eberhart, R.: Particle swarm optimization. IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
17. Khanra, A., Maiti, M.K., Maiti, M.: Profit maximization of TSP with uncertain parameters through a hybrid algorithm. In: Proceedings of the 4th international conference on frontiers in intelligent computing: theory and applications (FICTA), advances in intelligent systems and computing, vol. 404. Spinger, Berlin (2015)
18. Laporte, G., Nobert, Y.: Generalized traveling salesman through n sets of nodes: an integer programming approach. INFOR **21**, 61–75 (1983)
19. Laporte, G., Asef-Vaziri, A., Sriskandarajah, C.: Some applications of the generalized traveling salesman problem. J. Oper. Res. Soc. **47**, 1461–1467 (1996)
20. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. J. Evol. Comput. **10**(3), 281–295 (2006)
21. Liu, B.: Theory and Practice of Uncertain Programming. Springer, Heidelberg (2002)
22. Maiti, A.K., Maiti, M.K., Maiti, M.: Inventory model with stochastic lead-time and price dependent demand incorporating advance payment. Appl. Math. Model. **33**, 2433–2443 (2009)
23. Michalewicz, Z.: Genetic Algorithms + Data Structure = Evolution Programs. Springer, Berlin (1992)
24. Mondal, M., Maity, A.K., Maiti, M.K., Maiti, M.: A production-repairing inventory model with fuzzy rough coefficients under inflation and time value of money. Appl. Math. Model. **37**, 3200–3215 (2013)
25. Mohon, C.: Optimization in fuzzy-stochastic environment and its importance in present day industrial scenario. In: Proceedings on mathematics and its application in industry and business, Narosa Publishing House, India (2000)
26. Noon, C.E., Bean, J.C.: A Lagrangian based approach for the asymmetric generalized traveling salesman problem. Oper. Res. **39**, 623–632 (1991)

27. Pintea, C., Pop, P.C., Chira, C.: Reinforcing ant colony system for the generalized traveling salesman problem. In: Proc. of international conference bio-inspired computing-theory and applications (BIC-TA), Wuhan, China, Evolutionary Computing Section, pp. 245–252 (2006)
28. Pramanik, P., Maiti, M.K., Maiti, M.: A supply chain with variable demand under three level trade credit policy. Comput. Ind. Eng. **106**, 205–221 (2017)
29. Reinelt, G.: TSPLIB—a traveling salesman problem library. ORSA J. Comput. **3**, 376–384 (1991)
30. Renaud, J., Boctor, F.F.: An efficient composite heuristic for the symmetric generalized traveling salesman problem. Eur. J. Oper. Res. **108**(3), 571–584 (1998)
31. Saskena, J.P.: Mathematical model of scheduling clients through welfare agencies. J. Can. Oper. Res. Soc. **8**, 185–200 (1970)
32. Shi, X.H., Lianga, Y.C., Leeb, H.P., Lub, C., Wanga, Q.X.: Particle swarm optimization-based algorithms for TSP and generalized TSP. Inf. Process. Lett. **103**, 69–176 (2007)
33. Srivastava, S.S., Kumar, S., Garg, R.C., Sen, P.: Generalized traveling salesman problem through n sets of nodes. CORS J. **7**, 97–101 (1969)
34. Snyder, L.V., Daskin, M.S.: A random-key genetic algorithm for the generalized traveling salesman problem. Eur. J. Oper. Res. **174**, 38–53 (2006)
35. Wang, K.P., Huang, L., Zhou, C.G., Pang, W.: Particle swarm optimization for traveling salesman problem. In: International conference on machine learning and cybernetics, November 3, pp. 1583–1585 (2003)
36. Wu, C.G., Liang, Y.C., Lee, H.P., Lu, C.: A generalized chromosome genetic algorithm for generalized traveling salesman problems and its applications for machining. Phys. Rev. E **70**, 016701–113 (2004)
37. Yang, J., Shi, X., Marchese, M., Liang, Y.: An ant colony optimization method for generalized TSP problem. Prog. Nat. Sci. **18**, 1417–1422 (2008)
38. Yan, X., Zhang, C., Luo, W., Li, W., Chen, W., Liu, H.: Solve traveling salesman problem using particle swarm optimization algorithm. Int. J. Comput. Sci. Issues **9**(6), 264–271 (2012)
39. Zadeh, L.: Fuzzy sets. Inf. Control **8**, 338–356 (1965)
40. Zhao, X., Zhu, X.P.: Innovative genetic algorithm for solving GTSP. In: Second international conference on modeling, simulation and visualization methods, College of Computer Science and Engineering, Guangdong Institute of Science and Technology (2010)