European Journal of Operational Research 261 (2017) 819–834

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A hybrid Particle Swarm Optimization – Variable Neighborhood Search algorithm for Constrained Shortest Path problems

Yannis Marinakis[a,*], Athanasios Migdalas[b,c], Angelo Sifaleras[d]

[a] School of Production Engineering and Management, Decision Support Systems Laboratory, Technical University of Crete, University Campus, Chania 73100, Greece
[b] Department of Civil Engineering, Aristotle University of Thessalonike, 54124 Thessaloniki, Greece
[c] Industrial Logistics, Luleå Technical University, Luleå 97187, Sweden
[d] School of Information Sciences, Department of Applied Informatics, University of Macedonia, 156 Egnatias Str., Thessaloniki 54006, Greece

A R T I C L E I N F O

Article history:
Received 6 August 2014
Accepted 12 March 2017
Available online 16 March 2017

Keywords:
Particle Swarm Optimization
Variable Neighborhood Search
Expanding neighborhood topology
Constrained Shortest Path problem

A B S T R A C T

In this paper, a well known NP-hard problem, the Constrained Shortest Path problem, is studied. As efficient metaheuristic approaches are required for its solution, a new hybridized version of Particle Swarm Optimization algorithm with Variable Neighborhood Search is proposed for solving this significant combinatorial optimization problem. Particle Swarm Optimization (PSO) is a population-based swarm intelligence algorithm that simulates the social behavior of social organisms by using the physical movements of the particles in the swarm. A Variable Neighborhood Search (VNS) algorithm is applied in order to optimize the particles' position. In the proposed algorithm, the Particle Swarm Optimization with combined Local and Global Expanding Neighborhood Topology (PSOLGENT), a different equation for the velocities of particles is given and a novel expanding neighborhood topology is used. Another issue in the application of the VNS algorithm in the Constrained Shortest Path problem is which local search algorithms are suitable from this problem. In this paper, a number of continuous local search algorithms are used. The algorithm is tested in a number of modified instances from the TSPLIB and comparisons with classic versions of PSO and with other versions of the proposed method are performed. Also, the results of the algorithm are compared with the results of a number of metaheuristic and evolutionary algorithms. The results obtained are very satisfactory and strengthen the efficiency of the algorithm.

1. Introduction

Particle Swarm Optimization (PSO) is a population-based swarm intelligence algorithm that was originally proposed by Kennedy and Eberhart (1995). PSO belongs to the class of swarm intelligence algorithms, which are inspired by the social dynamics and the emergent behavior in socially organized colonies. PSO simulates the social behavior of social organisms by using the physical movements of the particles in the swarm. Its mechanism enhances and adapts to the global and local exploration. The basic PSO and its variants have been successfully applied for continuous optimization problems (Clerc & Kennedy, 2002; Eberhart & Shi, 2001; Janson & Middendorf, 2005; Parsopoulos & Vrahatis, 2002; Shi & Eberhart, 1998b). However, in the last years the PSO algorithm has, also, been used in discrete optimization problems (Consoli, Moreno-Perez, Darby-Dowman, & Mladenovic, 2008, 2010). Recent comprehensive surveys for PSO can be found in Banks, Vincent, and Anyakoha (2007), Banks, Vincent, and Anyakoha (2008), Poli, Kennedy, and Blackwell (2007). The Particle Swarm Optimization (PSO) is a very popular optimization method and its wide use, mainly during the last years, is due to a number of advantages that this method has compared to other optimization methods. Clerc and Kennedy (2002) proposed a constriction factor in order to prevent explosion, to ensure convergence and to eliminate the parameter that restricts the velocities of the particles. Usually, in PSO-based algorithms only one swarm is used. Recently, a number of works has been conducted that use more than one swarm either using the classic PSO (Brits, Engelbrecht, & Van Den Bergh, 2007; Niu, Zhu, He, & Wu, 2007; Tillett, Rao, Sahin, & Rao, 2005) or using some variations of the classic PSO like the method called TRIBES (Clerc, 2006). These methods have more exploration and exploitation abilities due to the fact that the different swarms have the possibility to explore different parts of the solution space.

* Corresponding author.
E-mail addresses: marinakis@ergasya.tuc.gr (Y. Marinakis), samig@civil.auth.gr, athmig@ltu.se (A. Migdalas), sifalera@uom.gr (A. Sifaleras).

http://dx.doi.org/10.1016/j.ejor.2017.03.031
0377-2217/© 2017 Elsevier B.V. All rights reserved.

In this paper, a significant Combinatorial Optimization problem, the Constrained Shortest Path problem, is solved. One of the main issues in the application of a Particle Swarm Optimization algorithm in combinatorial optimization problems is the fact that the basic equation of the Particle Swarm Optimization algorithm is suitable for continuous optimization problems and the transformation of this equation in the discrete space may cause lose of information and may, simultaneously, need a large number of iterations and the addition of a powerful local search algorithm in order to find an optimal solution. Usually, a path representation of the tour is applied but in the Constrained Shortest Path problem the use of this representation is proved not to be suitable for the efficient application of the Particle Swarm Optimization algorithm. Thus, in this paper a new representation based on the classic representation proposed by Shi and Eberhart (1998a) is presented which will help to speed up the process without losing information.

The algorithm is tested in a number of modified instances from the TSPLIB. We compare the algorithm with the results of two classic variants of the Particle Swarm Optimization algorithm, the inertia PSO (Shi & Eberhart, 1998a) and the constriction PSO (Clerc & Kennedy, 2002). Also, we compare the two versions of the proposed algorithm (the one with the expanding procedure based on the best particle and the other with the expanding procedure based on each particle separately) with a variant of the algorithm with the new equation but without the expanding procedure. Finally, the results of the proposed algorithm are compared with the results of nine metaheuristic and evolutionary algorithms. The rest of the paper is organized as follows: in Section 3 a review about neighborhood search topologies in PSO is added and, also, the new features of the proposed algorithm are given. In Section 2, a detailed description of the Constrained Shortest Path problem is presented. Section 4 presents an analytical description of the proposed algorithm. A detailed computational study based on the well known benchmark instances is presented in Section 5. Finally, the conclusions and future work follow in Section 6.

2. Constrained Shortest Path (CSP) problem

The Shortest Path (SP) problem is one of the oldest and its formulation often arises in combinatorial optimization (Dijkstra, 1959). The objective of the SP problem is to find the least cost path through a graph from a starting node to an ending node. A number of variants has been proposed which, mainly, add one or more constraints to the arcs. In this paper, the following variant of the Shortest Path problem is solved. Given a graph G for which each of its arcs is associated with two positive weights, cost and delay, we consider the problem of selecting a set of k arc-disjoint paths from a node s to another node t such that the total cost of these paths is minimum and that the total delay of these paths is not greater than a specified bound. This problem is called the constrained shortest arc-disjoint path problem (CSDP). If we would like to determine only one minimum cost path from node s to node t, then, the problem is called Constrained Shortest Path problem (CSP).

Computing CSP's is fundamental to several important network optimization problems in telecommunications and military aircraft management systems, just to name a few. In addition, in Formaneck and Cozzarin (2012) a Constrained Shortest Path problem is applied to training and technology adoption decisions by firms. An algorithm for the efficient solution of the Constrained Shortest Link-Disjoint Path Problem is presented in Xiao, Thulasiraman, and Xue (2006). A variant of the Constrained Shortest Path problem is the resource Constrained Shortest Path problem, which is the problem of finding the shortest path between two nodes on a network whenever the traversal of any arc or node consumes certain resources and the resources consumed along the path chosen must lie within given limits (both lower and upper limits) (Beasley & Christofides, 1989). In the literature, there is a number of papers that solve the resource constraint shortest path problem (Avella, Boccia, & Sforza, 2002). For a review about problems, software and applications please see Sifaleras (2013).

The CSP and its variants are used for air cargo planning and routing, flight planning, etc. (Cai, Zhang, Zhou, Cao, & Tang, 2012; Lefebvre, Puget, & Vilim, 2011; Liu, Zheng, & Cai, 2013; Mohemmed, Sahoo, & Geok, 2008; 2010; Royset, Carlyle, & Wood, 2009; Sura & Mahadevan, 2011; Wang, Lu, Zhang, Wang, & Yong Deng, 2014; Zhang, Zhang, Hu, Deng, & Mahadevan, 2013). The CSP consists of finding a path P from a start node $s \in N$ to an end node $e \in N$ that minimizes the total cost, subject to not exceeding a maximum resource consumption T. The CSP is known to be NP-hard even for the case of one resource. Thus, much research has been focused on designing efficient heuristic algorithms.

Let $G = (N, A)$ be a directed network with n nodes and m arcs, where N and A are the sets of nodes and arcs respectively. The CSP can be formulated as the following integer program (IP):

$$\min \sum_{(i_1,j_1)\in A} l_{i_1 j_1} r_{i_1 j_1} \tag{1}$$

$$s.t \sum_{k_1:(i_1,k_1)\in A} r_{i_1 k_1} - \sum_{j_1:(j_1,i_1)\in A} r_{j_1 i_1} = \begin{cases} 1, & if\ i_1 = 1 \\ 0 & if\ i_1 \neq 1, n \\ -1, & if\ i_1 = n \end{cases} \tag{2}$$

$$\sum_{(i_1,j_1)\in A} d_{i_1 j_1} r_{i_1 j_1} \leq T \tag{3}$$

$$r_{i_1 j_1} \in 0, 1, \ \forall (i_1, j_1) \ \in A \tag{4}$$

By $r_{i_1 j_1}$, $l_{i_1 j_1}$, and $d_{i_1 j_1}$ we denote the binary variables, the cost, and the delay associated with each arc (i_1, j_1), respectively. Furthermore, by T we denote the maximum value allowed for the delay.

Exact solution strategies for the CSP can be classified into one of two main categories, i.e., Dynamic Programing (DP) and Lagrangian relaxation. Methods based on DP are also known as label-setting or label-correcting algorithms (Joksch, 1966). Methods based on Lagrangian relaxation solve the integer programing formulation of the problem (Carlyle, Royset, & Wood, 2008). In Handler and Zang (1980) and Santos, Coutinho-Rodrigues, and Current (2007) two different Shortest Path methods have been proposed for the solution of the problem. Another exact algorithm has been presented in Lozano and Medaglia (2013). Since the CSP is NP-hard, several researchers have, also, tried other algorithmic approaches including approximate algorithms and metaheuristics. During the last years, it has become evident that a skilled combination of a metaheuristic with other optimization techniques, a so called hybrid metaheuristic (Blum, Puchinger, Raidl, & Roli, 2011), can provide a far more efficient behavior.

3. Local neighborhood topologies in PSO and new features of the proposed algorithm

In the PSO algorithm there are two kinds of population topologies: the global best (*gbest*) population topology and the local best (*lbest*) population topology (Engelbrecht, 2007). In the *gbest* PSO, the neighborhood for each particle is the entire swarm. The social network employed by the *gbest* PSO reflects the star topology in which all particles are interconnected. Thus, the velocities of each particle are updated based on the information obtained from the best particle of the whole swarm. In the *lbest* PSO, each particle has a smaller neighborhood. In this case, the network topology corresponds to the ring topology where each particle communicates with only a limited number of other members of the swarm. The communication is usually achieved using the particles' indices. Thus, if the size of the neighborhood is equal to three, the selected neighbors for the particle i are the particles $i-1$ and $i+1$. Thus, the velocities of each particle are updated based on the information obtained from the best particle of the neighborhood. The use of particle's indices for the creation of the neighborhood is preferred because it is very difficult and computationally expensive to calculate distances between all the particles to find the neighbors of each particle. Furthermore, if the indices are used, then, a particle may belong to more than one neighborhood having the possibility of spreading a good solution among different neighborhoods. Usually, the *gbest* PSO converges faster than the *lbest* PSO. On the other hand, the *lbest* PSO has larger diversity in its solutions and, thus, it is more difficult to be trapped in local minima (Engelbrecht, 2007).

In the past, a number of algorithms have been published using local neighborhood topologies but most of them are applied in global optimization problems or combinatorial optimization problems and not in routing problems as the ones studied in this paper. The first paper that designs and investigates neighborhood topologies (circle, wheel, star and random) was published by Kennedy (1999). In a following work Kennedy and Mendes (2002) proposed a number of population topologies. Mendes, Kennedy, and Neves (2004) proposed another local neighborhood topology, denoted as Full Informed Particle Swarm Optimization algorithm (FIPS). The first one that published an expanding neighborhood topology for the solution of global optimization problems was Suganthan (1999). Another two earlier adaptations of local neighborhood topologies are presented in Hu and Eberhart (2002) and Peram, Veeramachaneni, and Mohan (2003). In recent years, the adaptations of local neighborhood topologies in PSO have been increased. More precisely, in Wang, Sun, Li, Rahnamayan, and Pan (2013) a hybrid PSO algorithm is presented, which is denoted as DNSPSO where a diversity enhancing mechanism and neighborhood search strategies are used in order to achieve a trade-off between exploration and exploitation abilities. An improved version of this strategy is presented in Tran, Wu, and Wang (2013), where an enhanced Particle Swarm Optimization with diversity and neighborhood search (EPSODNS) approach was presented. In Liang, Qin, Suganthan, and Baskarr (2006) a Comprehensive Learning Particle Swarm Optimizer (CLPSO) was presented. An improvement of this method was proposed in Nasir et al. (2012), denoted as Dynamic Neighborhood Learning Particle Swarm Optimizer (DNLPSO), which uses a learning strategy whereby all other particles' historical best information is used to update a particle's velocity. In Wang et al. (2014) a multi-layer PSO method (MLPSO) consisting of global MLPSO and local MLPSO by increasing the swarm layers from two to multiple layers was proposed. Another PSO algorithm using a local search topology was presented in Li (2010) where a ring topology is used. In Lim and Isa (2014b), a Particle Swarm Optimization with increasing topology connectivity (PSO–ITC) was proposed to solve unconstrained single-objective optimization problems with continuous search space while in Lim and Isa (2014a) a PSO denoted as PSO with Adaptive Time-Varying Topology Connectivity (PSO–ATVTC) was proposed where the ATVTC module is used in order to balance the algorithm's exploration/exploitation searches by varying the particle's topology connectivity with time according to its searching performance. A time adaptive topology is proposed for constrained optimization problems in Bonyadi, Li, and Michalewicz (2014). In Jiang, Wang, and Wang (2013) an age-group topology Particle Swarm Optimization algorithm (PSOAG) is presented where the concept of age is used to measure the search ability of each particle in local area. The particles are divided in different age-groups by their age and particles in each age-group can only select the ones in younger groups or their own groups as their neighborhoods (Jiang et al., 2013). A cyclic neighborhood topology is presented in Maruta, Kim, Song, and Sugie (2013). An analysis of local best topologies is presented in Ghosh, Das, Kundu, Suresh, and Abraham (2012).

In Marinakis and Marinaki (2013b), a Particle Swarm Optimization with Expanding Neighborhood Topology (PSOENT) algorithm was presented and used for the solution of the Permutation Flowshop Scheduling Problem (PFSP). In this algorithm, the advantages of a local search neighborhood topology and of a global search neighborhood topology are used. The algorithm starts with a local search neighborhood topology where the neighborhoods for each particle are equal to 2 and in each iteration the neighborhood is increased (expanded) until it becomes equal to the number of particles. Then, if the maximum number of iterations has not yet been reached, the neighborhood is initialized, again, from a neighborhood equal to 2 and it follows the same procedure until the maximum number of iterations has been reached. Thus, there are no consecutive iterations in which the size of the neighborhood is the same. With this procedure each particle has more exploitation abilities by moving for a number of iterations in small swarms inside the whole swarm. Also, this procedure increased the exploration abilities as for a number of iterations all the particles are moving as a single swarm. By using an expanding neighborhood topology, each particle participated in each iteration in more than one swarm and, thus, the interchange of the information between the swarms is becoming easier.

In this paper, we modify the neighborhood topology presented in Marinakis and Marinaki (2013b). In Marinakis and Marinaki (2013b), a particle is affected by the best local neighborhood when the number of neighbors is not equal to the whole swarm, and by the best particle, otherwise. In the proposed algorithm, the Particle Swarm Optimization with combined Local and Global Expanding Neighborhood Topology (PSOLGENT), we give a different equation for the velocities where a fourth term has been added in the classic equation which represents the local neighbors. Thus, in the new equation the particle, except from a movement towards a new direction, a movement towards his previous best and a movement towards the global best of the swarm, could move towards the local best of its neighborhood. Another difference of the proposed algorithm from the algorithm proposed in Marinakis and Marinaki (2013b) is that in the PSOENT algorithm the neighborhood is expanding with the number of iterations, while in the proposed algorithm the neighborhood is expanding based on the quality of the solutions. We test two different alternatives. In the first one, the neighborhood is expanding when the optimal particle is not improved for a specific number of iterations (it_{num}). In the second alternative, each particle has its own neighborhood. Thus, while all particles begin with the same neighborhood topology when a solution (particle's position) of a particle is not improved for a consecutive number of iterations, then, only its neighborhood is expanding. With this strategy there is a possibility of having different neighborhood topologies for each particle which is another novelty of the proposed algorithm. This idea of changing the neighborhood topology of the swarm not with the number of iterations but when the swarm cannot improve the global (first alternative) or personal best solution (second alternative) is inspired by the basic idea of changing a neighborhood when the algorithm is trapped in a local optimum in the Variable Neighborhood Search (VNS) algorithm (Hansen & Mladenović, 2001). Thus, as in VNS algorithm the neighborhood is expanding in order to find a better local optimum, in the proposed algorithm the search for a better direction that the particle will move in its neighborhood is expanding in order to search in a larger neighborhood when the particle (position of the particle) is trapped in a local optimum. This incorporation of one of the basic characteristics of Variable Neighborhood Search algorithm in the Particle Swarm Optimization gave a powerful version of PSO algorithm. Another characteristic of VNS algorithm that is incorporated in the PSO is the reinitialization of the search for the local best neighborhood. In the proposed algorithm, when the number of neighbors becomes equal to the number of particles, then, the search of the local best neighborhood is reinitialized from a very small neighborhood. In each iteration of the algorithm the search is realized from a different point (shaking procedure of the VNS) as the current position of each particle is different in each iteration.

In addition, the VNS algorithm is also applied in order to optimize the position of the particles. The basic idea of the VNS algorithm is the successive search in a number of neighborhoods of a solution. It should be noted that in a Particle Swarm Optimization algorithm with the term "neighborhood" it is meant the topology of the swarm (the particles' position that affect another particle's position in the process of finding the optimum), while in a Variable Neighborhood Search algorithm with the term "neighborhood" it is meant different numbers of local search algorithms. In the rest of the paper when the term "neighborhood" is used together with the term "topology" we will refer to the Particle Swarm Optimization part of the algorithm and when it is used together with the term "structure" we will refer to the local search algorithms that are used inside the Variable Neighborhood Search algorithm. Another issue in the application of the VNS algorithm in the Constrained Shortest Path problem is which local search algorithms are suitable for this problem. More precisely, could it be used an algorithm like 2-opt, which is the most usually used algorithm for combinatorial, and especially routing, problems? As it is proved from the results which will be analyzed in more details in Section 5.2, the answer is that 2-opt or 1–0 relocate or 1–1 exchange are not suitable for the problem. This is due to the fact that the VNS algorithm is applied in a solution which comes from a continuous optimization algorithm, the PSO, after its transformation to discrete space as it will be explained in Sections 4.1 and 4.2. But, then, which local search will be used for the problem? In this paper, we use a number of continuous optimization local search procedures which are directly connected with the two novel parts of the PSO algorithm, the equation of velocities and the expanding neighborhood topology of the particles. Thus, the interaction of the VNS algorithm and the Particle Swarm Optimization algorithm led us in a hybridized algorithm.

4. Particle Swarm Optimization with combined Local and Global Expanding Neighborhood Topology (PSOLGENT)

In this section, the proposed hybrid Particle Swarm Optimization algorithm, the PSO with combined Local and Global Expanding Neighborhood Topology (PSOLGENT), for the solution of the CSP is given. In a PSO algorithm, initially, a set of particles is created randomly where each particle corresponds to a possible solution. Each particle has a position in the space of solutions and moves with a given velocity. One of the key issues in designing a successful PSO for the Constrained Shortest Path problem is to find a suitable mapping between Constrained Shortest Path solutions and particles in PSO (see Section 4.1). The position of each particle is represented by a x_{ij}, $i = 1, 2, \ldots, NP$ and $j = 1, 2, \ldots, N$ (NP is the population size and N is the number of nodes), and its performance is evaluated on the predefined fitness function ($f(x_{ij})$) (see Section 2, the Eq. (1) and Section 4.1 for a complete enumeration of the fitness function for the specific problem). The velocity v_{ij} represents the changes that will be made to move the particle from one position to another. Where the particle will move depends on the dynamic interaction of its own experience and the experience of the whole swarm. There are three possible directions that a particle can follow: to follow its own path, to move towards the best position it had during the iterations ($pbest_{ij}$) or to move to the best particle's position ($gbest_j$). In this paper, a new velocity equation is proposed where a fourth direction that a particle can follow (the best local neighbor of each particle ($lbest_{ij}$)) is added. Initially, the routes of each particle are created with random node sequences and the particles' velocities are initialized with zeros.

In the classic Particle Swarm Optimization (Kennedy & Eberhart, 1995), the equation of velocities is given from the following equation:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 rand_1 (pbest_{ij} - x_{ij}(t)) + c_2 rand_2 (gbest_j - x_{ij}(t)) \quad (5)$$

where c_1 and c_2 are the acceleration coefficients, $rand_1$ and $rand_2$ are two random variables in the interval [0, 1]. The acceleration coefficients c_1 and c_2 control how far a particle will move in a single iteration. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement towards target regions (Kennedy & Eberhart, 1995).

In this paper, except of the proposed equation of velocities that it will be described in the following, two other variants of the Particle Swarm Optimization are used for comparisons purposes. These variant use the following equations of velocities.

The first one is the Inertia Particle Swarm Optimization (iPSO) (Shi & Eberhart, 1998a):

$$v_{ij}(t+1) = w v_{ij}(t) + c_1 rand_1 (pbest_{ij} - x_{ij}(t)) + c_2 rand_2 (gbest_j - x_{ij}(t)) \quad (6)$$

The difference of this equation from Eq. (5) is the use of the inertia weight w. Initially, the inertia weight has a large value in order to affect as much as possible the velocity and, subsequently, the position of each particle. The inertia weight is decreased during the iterations as it is possible the particles to converge very fast in a swarm and a large value of the velocity may dissolve this swarm. Finally, in the last iterations the inertia weight is almost equal to zero as it is preferred not to affect at all in the velocity and in the position of each particle. The inertia weight w is, also, used to control the convergence behavior of the PSO. A number of different alternatives for the definition of w has been proposed. These alternatives vary from constant values to different ways of increasing or decreasing of w during the iterations. In this paper, in order to exploit more areas in the solution space, the inertia weight w is updated according to the following equation:

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times t \quad (7)$$

where w_{max}, w_{min} are the maximum and minimum values of inertia weight.

The second variant is the Constriction Particle Swarm Optimization (cPSO) (Clerc & Kennedy, 2002):

$$v_{ij}(t+1) = \chi (v_{ij}(t) + c_1 rand_1 (pbest_{ij} - x_{ij}(t)) + c_2 rand_2 (gbest_j - x_{ij}(t))) \quad (8)$$

where

$$\chi = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|} \text{ and } c = c_1 + c_2, c > 4 \quad (9)$$

The difference of this variant from the two previous variants is the use of a limiting factor (χ), called constriction factor. The constriction factor is used in order to prevent explosion, to ensure convergence of the algorithm and to eliminate the factors that limit the velocities of the particles.

In this paper, a fourth term in the equation of velocities is added that represents the interaction of each particle with its local best neighbor as it is described in Section 3. Thus, the proposed equation of PSO becomes (PSOLGENT):

$$v_{ij}(t+1) = \chi_1 (v_{ij}(t) + c_1 rand_1 (pbest_{ij} - x_{ij}(t)) \quad (10)$$

$$+c_2 rand_2(gbest_j - x_{ij}(t)) + c_3 rand_3(lbest_{ij} - x_{ij}(t)))$$

where c_3 is the acceleration coefficient, $rand_3$ is a random parameter in the interval [0, 1], t is the iterations? counter and χ_1 is the constriction factor, i.e.:

$$\chi_1 = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|} \text{ and } c = c_1 + c_2 + c_3, c > 4 \tag{11}$$

A particle's best position ($pbest_{ij}$) in a swarm is calculated from the equation:

$$\mathbf{pbest}_i = \begin{cases} \mathbf{x_i(t+1)}, & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{x}_i(t)) \\ \mathbf{pbest}_i, & \text{otherwise} \end{cases} \tag{12}$$

The optimal position of the whole swarm in the CSP at time t is calculated from the equation:

$$\mathbf{gbest} \in \{\mathbf{pbest}_1, \mathbf{pbest}_2, \ldots, \mathbf{pbest}_{NP}\} | f(\mathbf{gbest}) = min\{f(\mathbf{pbest}_1), f(\mathbf{pbest}_2), \ldots, f(\mathbf{pbest}_{NP})\} \tag{13}$$

and the $lbest_{ij}$ in Eq. (10) is calculated as follows:

$$\mathbf{lbest}_i \in \{Neigh_i | f(\mathbf{lbest}_i) = min\{f(\mathbf{x}_i)\}, \forall x \in Neigh_i\} \tag{14}$$

and the neighbor $Neigh_i$ is defined by Engelbrecht (2007):

$$Neigh_i = \{\mathbf{pbest}_{i-n_{Neigh_i}}, \mathbf{pbest}_{i-n_{Neigh_i}+1}, \ldots, \mathbf{pbest}_{i-1}, \mathbf{pbest}_i, \mathbf{pbest}_{i+1}, \ldots, \mathbf{pbest}_{i+n_{Neigh_i}}\} \tag{15}$$

In the proposed Particle Swarm Optimization with combined Local and Global Expanding Neighborhood Topology (PSOLGENT) algorithm, an issue that is very important to evaluate is the way the neighbors of each particle are calculated as it is described in Section 3. For comparison purposes we use, also, a variant of the proposed algorithm with static neighborhood topology (PSOLGNT).

The position of a particle changes using the following equation:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{16}$$

As the problem is a combinatorial optimization problem and the positions of the particles are in continuous form, we use the following sigmoid function for the transformation of the solutions (Shi & Eberhart, 1998a):

$$sig(x_{ij}) = \frac{1}{1 + exp(-x_{ij})} \tag{17}$$

and, then, the position is calculated by:

$$s_{ij}(t+1) = \begin{cases} 1, & \text{if } rand_4 < sig(x_{ij}) \\ 0, & \text{if } rand_4 \geq sig(x_{ij}) \end{cases} \tag{18}$$

where $rand_4$ is a random number in the interval [0, 1] and s_{ij} is the position of each particle in discrete form. A Variable Neighborhood Search strategy is used in order to improve the solutions (particles' positions) produced from the Particle Swarm Optimization algorithm (see Section 4.2). In each iteration of the algorithm the optimal solution of the whole swarm and the optimal solution of each particle are kept. The algorithm stops when a maximum number of iterations has been reached. A pseudocode of the proposed algorithm is presented in Table 1.

4.1. Example of a solution and path representation

The PSO algorithm works using continuous values. Thus, initially, the solutions are randomly placed in the solution space and, then, in order to calculate the fitness function and the feasibility of a solution, the values are transformed into discrete space by using the Eqs. (17) and (18). In all iterations of the algorithms we have two different vectors that represent the solution. The one vector has continuous values and it is used in all steps of the algorithms and the other vector has discrete values and it is used only for the calculation of the fitness function.

Table 1
Particle Swarm Optimization (PSOLGENT) for the Constrained Shortest Path problem.

```
Initialization
Select the number of particles in each swarm
Initialize the position and velocity of each particle
Calculate the initial fitness function of each particle
Find the best position of each particle
Find the best particle of the entire swarm
Find the local best neighborhood for each particle
Main phase
Do while the maximum number of iterations has not be reached
    Calculate the velocity of each particle
    Calculate the new position of each particle
    Convert particles' positions in discrete space
    Calculate the new fitness function of each particle
    Improve the solutions using VNS
    Update the best position of each particle
    Find the best particle of the whole swarm
    Find the local best neighborhood for each particle
Enddo
Return the best particle
```

For example, if we have the following solution in the continuous space:

0.87 −3 2.1 2.75 −0.048 0.35 −1.29 1.11

with the use of Eq. (17) the values are transformed in the interval [0,1] as follows:

0.704 0.047 0.890 0.939 0.488 0.586 0.215 0.752

and, then, for each node a different random number ($rand_4$) in the (0,1) is created, for example:

0.576 0.187 0.918 0.876 0.325 0.634 0.129 0.359.

Finally, as all the solutions are represented with 0 and 1 values where a zero value means that the node does not belong to the shortest path while a value equal to 1 means that this node belongs to the path, from Eq. (18) the vector is transformed in the discrete space as follows:

1 0 0 1 1 0 1 1

which means that the shortest path is the solution

1 4 5 7 8

The starting and ending nodes have always values equal to 1 and as the distances are Euclidean we delete any direct arc from the starting node to the ending node as we will explain in more detail in Section 5.1.

We do not know yet the fitness function (cost of the solution) and the feasibility of the solution for this example. Initially, in order to test the feasibility of the solution, we have two constraints (Eqs. (2) and (3)). The first constraint is never violated through the structure of the solution vector in discrete space. For the second constraint we have to calculate the quantity $d_{i_1 j_1} r_{i_1 j_1}$ for all activated (value equal to 1) nodes in the solution. Thus, in our example we have to calculate $d_{14}r_{14} + d_{45}r_{45} + d_{57}r_{57} + d_{78}r_{78}$. This quantity has to be less than a value T. If this condition holds, then, the solution is feasible and the fitness function is calculated, otherwise, the solution is not feasible and a penalty function is added in the fitness function in such a way that this solution will never be selected as a best solution. The reason that we did not reject a non feasible solution is that we would not like to repeat the procedure of the construction of a solution a number of times in an iteration which will slow down the algorithm. The fitness function is calculated by the Eq. (1). In our example a complete enumeration of the fitness function is $l_{14}r_{14} + l_{45}r_{45} + l_{57}r_{57} + l_{78}r_{78}$.

In the local search phase of the algorithm other sequences of the selected nodes could be examined. The reason that we begin from a sequence of nodes in which the nodes are used in ascending order is that this representation is the most efficient for the PSO algorithm. However, the sequence in an ascending order could produce a not good solution, thus, in the local search phase a rearrangement of the selected nodes could be performed (keeping the starting and ending nodes the same) and other sequences of the nodes (for example 1–7–5–4–8 and 1–5–7–4–8) could be examined.

4.2. Variable Neighborhood Search (VNS)

A Variable Neighborhood Search (VNS) (Hansen & Mladenović, 2001; Hansen, Mladenović, & Moreno-Pérez, 2010) algorithm is applied in order to optimize the particles' position. The search is applied either with random or with a more systematical manner in order the solution to escape from a local minimum. This method takes advantage of the fact that different local search algorithms will lead to different local minima. The first issue that we have to answer in our problem is which are the local search strategies that will combine the neighbourhood structure. Initially, a number of classic local search algorithms for combinatorial optimization problems (2-opt; 1–0 relocate; and 1–1 exchange) are tested independently and, afterwards, they are tested within three different VNS frameworks by using the inertia PSO. As it will be presented analytically in Section 5.2, the local search algorithms and the VNS algorithm did not improve the results significantly and in some cases they did not improve the results at all. This is due to the fact that the Particle Swarm Optimization algorithm is a continuous optimization algorithm and the transformation in the discrete space mainly lead to lose information and to, probably, destroy a good solution and, thus, a discrete local search does not lead to improvement of the solutions. This is the main reason why there are not so many applications of PSO algorithm in routing problems. An efficient way that gave very good results to a number of problems (Probabilistic Traveling Salesman Problem, Marinakis and Marinaki, 2010b; Vehicle Routing Problem, Marinakis and Marinaki, 2010b; 2013a; Marinakis, Marinaki, and Dounias, 2010; and Location Routing Problem, Marinakis & Marinaki, 2008) is to replace the equation of positions with a Path Relinking (Glover, Laguna, & Marti, 2003) and to change the role of equation of velocities. In the last case, the 2-opt, the 1–0 relocate, the 1–1 exchange and every other local search algorithm of this kind could be used in a VNS framework effectively. However, in the problem studied in this paper, as the path representation of the route is not used in the representation of the solutions, the path relinking algorithm cannot be used.

In order to avoid the losing of information, it is decided to retain the continuous values in local search phase and to use local search algorithms suitable for continuous optimization problems. Six different local search equations are used (LS1,…, LS6). All the algorithms are applied for a number of iterations (ls_{num}). The first one uses a transformation of the solution inside the solution space. The second one combines the current solution with the global best particle and the third one combines the current solution with the local best particle. The fourth is a combination of the current solution, the global best and the local best particle. The fifth and the sixth are kinds of crossover with the local best and the global best particle, respectively. The methods used are described with the following equations, respectively:

$$\text{LS1: } x_{ij}(t_1+1) = rand_5 * x_{ij}(t_1) \tag{19}$$

$$\text{LS2: } x_{ij}(t_1+1) = rand_6 * gbest_j + (1-rand_6) * x_{ij}(t_1) \tag{20}$$

$$\text{LS3: } x_{ij}(t_1+1) = rand_7 * lbest_{ij} + (1-rand_7) * x_{ij}(t_1) \tag{21}$$

$$\begin{aligned}\text{LS4: } & x_{ij}(t_1+1) \\ &= rand_8 * rand_9 * gbest_j \\ &\quad + rand_8 * (1-rand_9) * lbest_{ij} + (1-rand_8) * x_{ij}(t_1)\end{aligned} \tag{22}$$

$$\text{LS5: } x_{ij}(t_1+1) = \begin{cases} lbest_{ij}, & \text{if } rand_{10} \le 0.5 \\ x_{ij}(t_1), & \text{otherwise.}\end{cases} \tag{23}$$

$$\text{LS6: } x_{ij}(t_1+1) = \begin{cases} gbest_j, & \text{if } rand_{11} \le 0.5 \\ x_{ij}(t_1), & \text{otherwise.}\end{cases} \tag{24}$$

where t_1 is the local search iteration number, $rand_6, \dots, rand_{11}$ are random numbers in the interval [0,1] and $rand_5$ is a random number in the interval [−1,1]. Two different approaches of the VNS algorithm were selected. The first one is the classic version where the VNS algorithm begins from a local search algorithm and when a local optimum is found the algorithm proceeds to the next local search algorithm, while the second is a Sequential version of VNS (SVNS), where in each iteration all the local searches are called sequentially. In each approach, two versions are selected, in the one (VNS1 and SVNS1) only the continuous local search algorithms are used while in the other (VNS2 and SVNS2) both continuous and discrete local search algorithms are used. Also, three versions with only discrete local search algorithms are used (DVNS1–DVNS3).

5. Computational study

5.1. Experimental design

The algorithm (implemented in Fortran 90) was tested on the 21 (modified) benchmark instances from the TSPLIB. We transformed the instances in order to be suitable for the CSP problem. The number of nodes is between 51 and 439. We have two end-nodes for each instance, the first and the last of each file. There is no connection between the first and the last node. There is only one direction between two nodes. The cost is the distance between two nodes. We added a random value for each arc which is the delay of each arc. Finally, we added a value T different for each instance which is dependent from the number of nodes and the delay values.

The parameters of the proposed algorithm are selected after thorough testing. A number of different alternative values were tested and the ones selected are those that gave the best computational results concerning both the quality of the solution and the computational time needed to achieve this solution. The selected parameters for all algorithms are presented in Table 2. In this table, in the places where there are not any values, these parameters are not used in the corresponding algorithm. In the first column the name of the parameter is given.

5.2. Computational results

Initially, in the inertia PSO, a number of classic local search algorithms for combinatorial optimization problems are tested (the 2-opt, the 1–0 relocate and the 1–1 exchange) independently and,

Table 2
Parameters for all algorithms.

	iPSO	cPSO	PSOLGNT	PSOLGENT1	PSOLGENT2
Particles	100	100	100	100	100
Iterations	1000	1000	1000	1000	1000
ls_{num}	10	10	10	10	10
it_{num}	–	–	–	10	10
Neighbors	–	–	3	3–99	3–99
c_1	2.0	2.05	1.35	1.35	1.35
c_2	2.0	2.05	1.35	1.35	1.35
c_3	–	–	1.40	1.40	1.40
w_{max}	0.9	–	–	–	–
w_{min}	0.01	–	–	–	–

afterwards, these three algorithms were used in three different VNS frameworks. In Table 3, the twenty-one instances described previously are presented. We tested these local search algorithms with the use of the inertia PSO as we would like to test their performance using the most classic variant of Particle Swarm Optimization. From this table, it is observed that these local search algorithms are not suitable for the problem. The results without local search are, in some instances, similar to the results with a local search but in some instances are even better than the results with a local search. This is due to the fact that the Particle Swarm Optimization algorithm is a continuous optimization algorithm and the transformation in the discrete space, mainly, leads to losing information and to, probably, destroying a good solution and, thus, a discrete local search does not lead to improvement of the solutions. It is very difficult to hybridize one continuous optimization algorithm and a local search which is suitable for combinatorial optimization problems. There is a number of combinatorial optimization problems in the literature (Marinakis & Marinaki, 2008; 2010a; 2010b) but all of them are using a path representation of the solution and other features which give a more effective hybridization. However, as the path representation is not suitable for the studied problem, it is concluded from Table 3 that the use of only discrete local search algorithms, either alone or in combination in a VNS framework, will not help us for the research. Thus, a number of continuous local search algorithms are developed (see Section 4.2) and from these algorithms different frameworks of VNS are constructed.

In Table 4 and in Figs. 1–3 the results of the five variants of Particle Swarm Optimization algorithm without any local search or Variable Neighborhood Search algorithm are presented. In all tables and figures of this section with iPSO is denoted the inertia PSO, with cPSO is denoted the constriction PSO, with PSOLGNT is denoted the proposed algorithm with one of the two basic features, i.e. only the new equation of velocities and with static number of local neighborhoods, with PSOLGENT1 is denoted the proposed algorithm with both new features activated and the expanding procedure depending from the optimal particle, while with PSOLGENT2 is denoted the proposed algorithm with both new features activated and the expanding procedure depending from the personal best position of each particle independently. All figures of this section present how the solution of the optimal particle is improved during the iterations. The worst of all algorithms, concerning both the quality of the solution and the iterations needed to find its best solution, is the inertia PSO. Another interesting outcome of these figures is that in the constriction PSO, although the best solution is improved fast in the first 200 iterations, in the remaining it is trapped in a local optimum which in two figures (Figs. 1 and 2) is worst even from the solution found using the inertia PSO. The constriction PSO found a competitive solution compared to the solutions of PSOLGNT and PSOLGENT2 in instance ts225.

The three proposed variants (PSOLGNT, PSOLGENT1, PSOLGENT2) perform better than the other two (iPSO, cPSO) in all instances. We could not say that one of the proposed methods outperforms clearly the other two methods, as in each figure one of them finds the optimal solution (in Fig. 1 the PSOLGENT2, in Fig. 2 the PSOLGNT and in Fig. 3 the PSOLGENT1). In conclusion, we could say that the PSOLGENT1 performs slightly better than the other two algorithms both in the best solution found and in the convergence speed to its best solution.

Table 4
Computational results without local search.

	a280	pr439	lin105	kroA200	ts225
iPSO	814.16	49574.99	1750.7	10306.54	69380.99
cPSO	919.35	55432.42	1742.875	4356.84	53307.67
PSOLGNT	733.5	36445.25	1742.875	4311.05	51661.29
PSOLGENT1	627.74	44405.02	1742.875	3563.01	31657.91
PSOLGENT2	596.73	43195.92	1742.875	4176.7	50679.23

Table 3
Computational results using inertia PSO.

	Without local search	2-opt	1–0 relocate	1–1 exchange	Classic VNS	SVNS	RVNS
eil51	16.06	16.06	16.06	16.06	16.06	16.06	16.06
eil76	25.49	25.49	25.49	25.49	25.86	26.43	25.49
kroA100	2643.82	2643.82	2644.56	2644.56	2643.82	2649.67	2643.82
kroB100	1248.78	1248.78	1248.78	1248.78	1270.49	1269.49	1248.78
kroC100	1650.46	1653.27	1650.46	1650.46	1650.46	1653.27	1653.27
kroD100	2718.8	2720.38	2720.38	2718.8	2720.38	2720.38	2718.8
kroE100	3499.36	3498.77	3497.39	3502.61	3497.39	3499.36	3497.39
eil101	15.81	16.63	26.54	26.54	15.81	15.81	15.81
lin105	1744.12	1914	1743.3	1856.91	1742.89	1742.87	1742.89
pr144	11590.19	11161.6	17185.33	20535.44	11061.19	11456.08	11132.76
kroA150	1474.44	1854.23	1835.66	2162.44	1382.98	1382.63	1382.63
pr152	13719.44	19992.6	21176.07	20234.59	13719.13	13719.19	16767.91
kroA200	6990.5	6052.03	11283.61	10222.58	7348.06	6978.04	8152.19
kroB200	4666.62	8965.21	15120.65	5479.43	6089.13	7680.26	8542.98
ts225	51504.01	38735.87	33053.6	58715.03	45063.23	46072.51	48883.41
pr226	29769.61	32862.08	31366.39	42188.44	28908.95	26923.78	30875.91
Gil262	328.85	393.13	1060.05	873.1	845.36	692.8	798.51
pr264	20592.55	14825.74	17532.44	18712.27	18122.91	18549.12	18007.49
A280	829.57	841.48	818.64	893.51	929.38	828.81	550.19
pr299	23467.86	24556.92	21627.41	26305.97	22889.89	23437.74	20576.8
pr439	44653.69	40316.01	49682.03	56850.9	48176.53	39724.18	43018.71

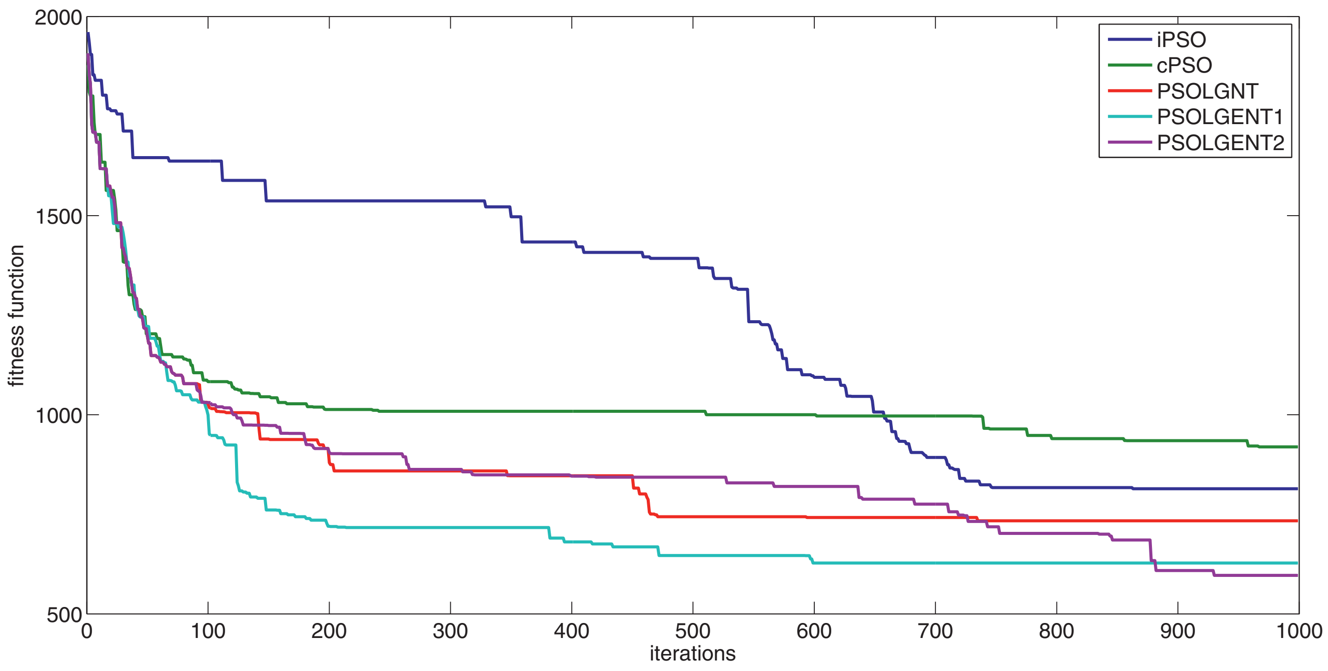

Fig. 1. Results of the five variants of PSO without local search for the instance a280.

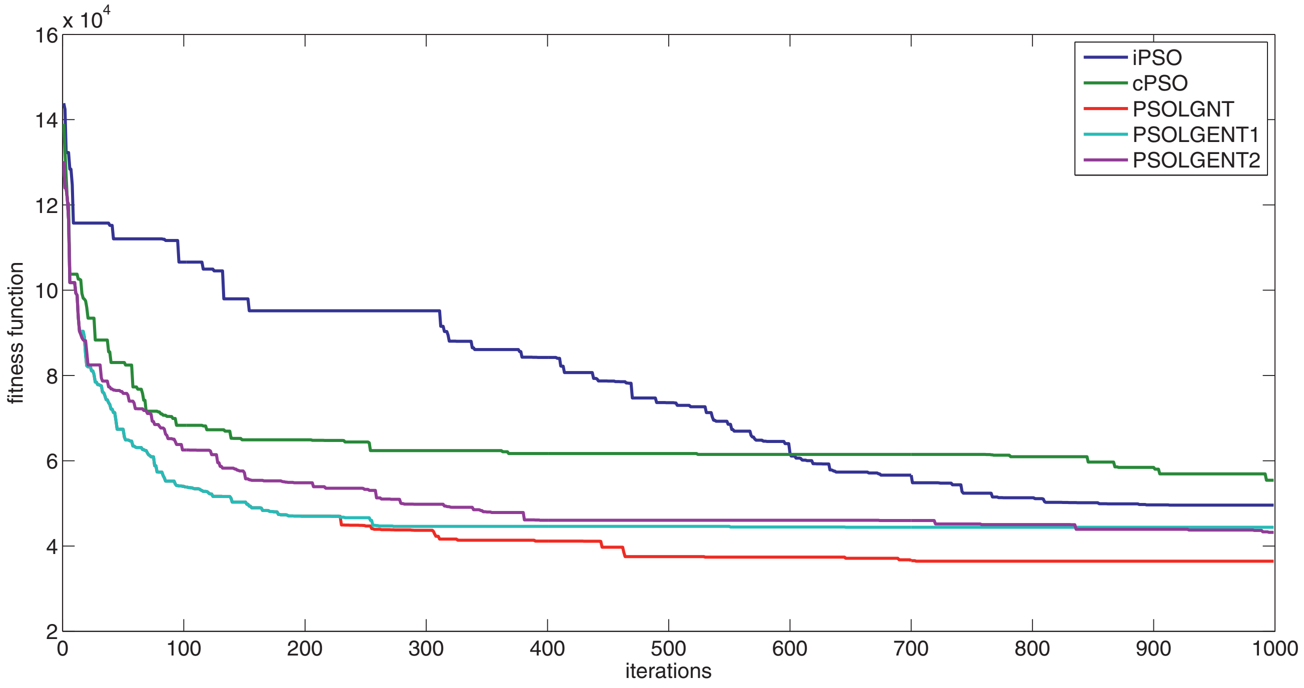

Fig. 2. Results of the five variants of PSO without local search for the instance pr439.

In Table 5 the analytical results for a specific instance, the a280, for all variants of PSO and all combinations with simple local search algorithms and Variable Neighborhood Search methods are presented. In this table, in the places where no values exist it was impossible to run this variant of PSO with this combination of local search algorithms. This is due to the fact that the local search algorithms from LS2 to SVNS2 need always to find the local best particle of each particle which is impossible in the classic versions of iPSO and cPSO. In VNS1 and SVNS1 only the continuous local search is used while in VNS2 and SVNS2 both continuous and discrete local searches are used. From Table 5 it is clear that when a continuous local search algorithm is used the results are improved significantly, especially, compared to the results found without local search (WLS first row of the table). Also, the results of VNS using discrete local search algorithms are worse than the results obtained when only a single continuous local search algorithm is used. A very interesting outcome is concluded from this table when each row of the table is observed. In each row the results of all PSO variants with the use of the same local search algorithm are given. For example, when all algorithms use each one of the DVNS local search procedures the algorithm that found better results is the iPSO variant. However, the results of the iPSO with

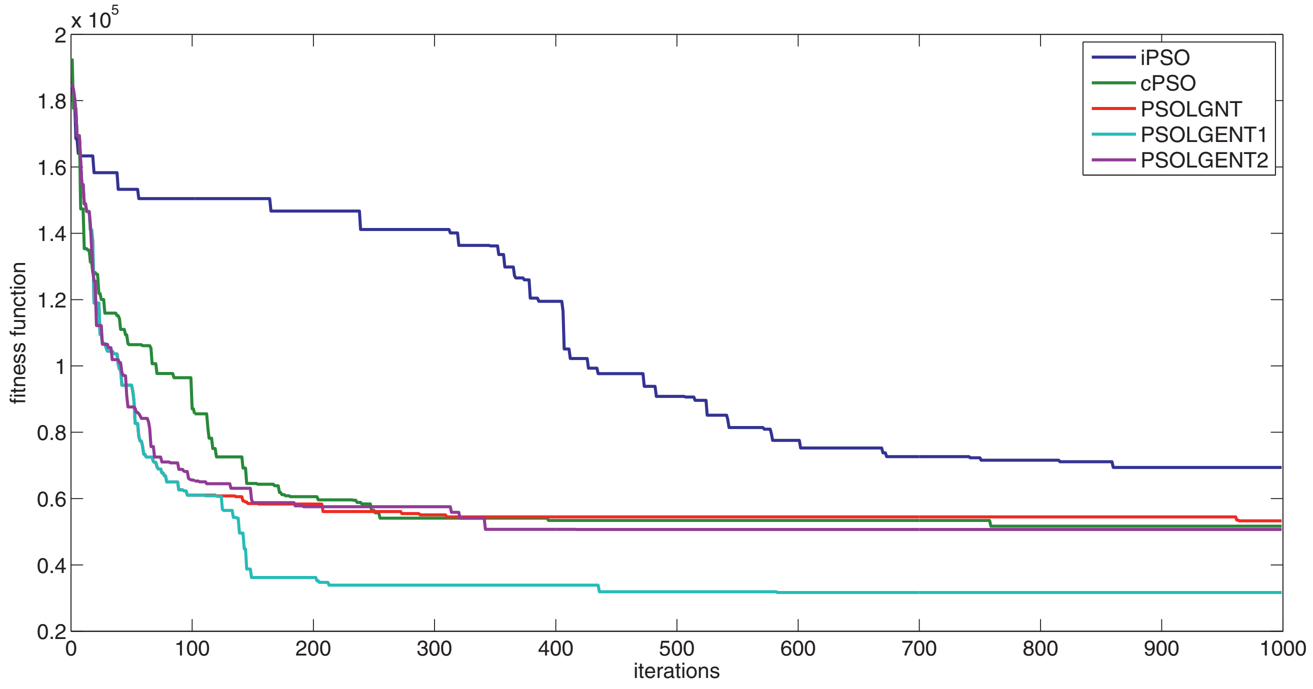

Fig. 3. Results of the five variants of PSO without local search for the instance ts225.

Table 5
Computational results for the instance a280.

Method	iPSO	cPSO	PSOLGNT	PSOLGENT1	PSOLGENT2
WLS	814.16	919.35	733.5	627.74	596.73
2-opt	969.53	1298.51	926.97	1025.27	911.73
1–0 relocate	1334.2	1343.03	985.66	957.33	912.49
1–1 exchange	93.47	650.66	342.25	214.37	380.44
DVNS1	841.91	1186.05	924.62	966.02	941.25
DVNS2	669.83	1339.15	873.18	988.05	931.59
DVNS3	792.94	1286.14	817.09	863.4	850.07
LS1	1004.4	1093.47	630.2	731.07	700.98
LS2	–	–	28.94	28.94	34.08
LS3	–	–	28.94	28.94	28.94
LS4	–	–	58.28	28.94	28.94
LS5	–	–	646.53	675.01	466.26
LS6	–	–	466.26	675.01	646.53
VNS1	–	–	54.8	28.94	34.08
VNS2	–	–	77.68	58.28	28.94
SVNS1	–	–	28.94	28.94	28.94
SVNS2	–	–	28.94	28.94	28.94

any of these local search procedures are worst than the best solution found from other PSO variants using a different local search procedure. Thus, the iPSO algorithm using discrete VNS procedures gives better results than the other PSO variants but is stuck in a bad local optimum (the best value found by iPSO with DVNS3 is 792.94 while the best solution found is 28.94 that has been found by many others combinations of PSO with a different local search algorithm). In this algorithm (iPSO) as it was explained earlier it was impossible to apply a continuous local search algorithm. The use of a local search procedure in the other PSO variants either as a combination of only continuous local search procedures or as a combination of continuous and discrete local search procedures helped the algorithm to escape from this local optimum and to find much better local optimums.

In Figs. 4–6 the results of all VNS variants for the three proposed PSO implementations and for the specific instance a280 are presented. As it can be seen when only discrete local search algorithms are used (DVNS1, DVNS2 and DVNS3) the results do not converge at all for the specified number of iterations. The other four versions converge to a very good solution, in most of the cases to the same solution, but when the SVNS1 or SVNS2 version are used the convergence is succeeded faster. The reason that the continuous local search algorithms perform better than the discrete local search algorithms is that the first are applied in the continuous vector and, thus, take advantage of the improvement of the solutions using PSO while the latter are applied in the transformed discrete vector which may lead to lose information and to, probably, destroy a good solution. Thus, a discrete local search does not lead to an improvement of the solutions or to a fast convergence to a good solution. The sequential versions of VNS perform better than the classic versions of VNS, because as it was shown from the results, the fact that in one iteration all local search algorithms are applied sequentially helps the algorithm to find faster a better solution than the other case, where each local search algorithm is applied until it is trapped to a local optimum, and, then, the algorithm proceeds to a different one.

In Table 6 and in Figs. 7–9 the analytical results for a specific instance, the pr439, using all continuous local search algorithms and Variable Neighborhood Search implementations which incorporate a continuous local search algorithm are presented. Also, the results without any local search are presented. As it can be seen from these figures, the LS5 and LS6, where the curves are almost identical, do not give good results at all and their results are competitive only with the case where local search algorithm is not applied (WLS). All the other local search algorithms find in almost all runs the same best solution, or a solution near to the best solution. The SVNS1 and SVNS2 variants converge faster than the other variants. The LS2–LS4 variants give competitive results compared to the results of the VNS1 and VNS2 algorithms. This proves the effectiveness of these local search algorithms. Especially, the LS4 algorithm performs better than other local search algorithms in all runs except from the SVNS1 and SVNS2 variants.

In Table 7 the results of the 21 instances in the three proposed versions of PSO using either no local search at all or the two variants of VNS that gave better results are presented. As we can see in the instances with small number of nodes (from 51 to 105) all

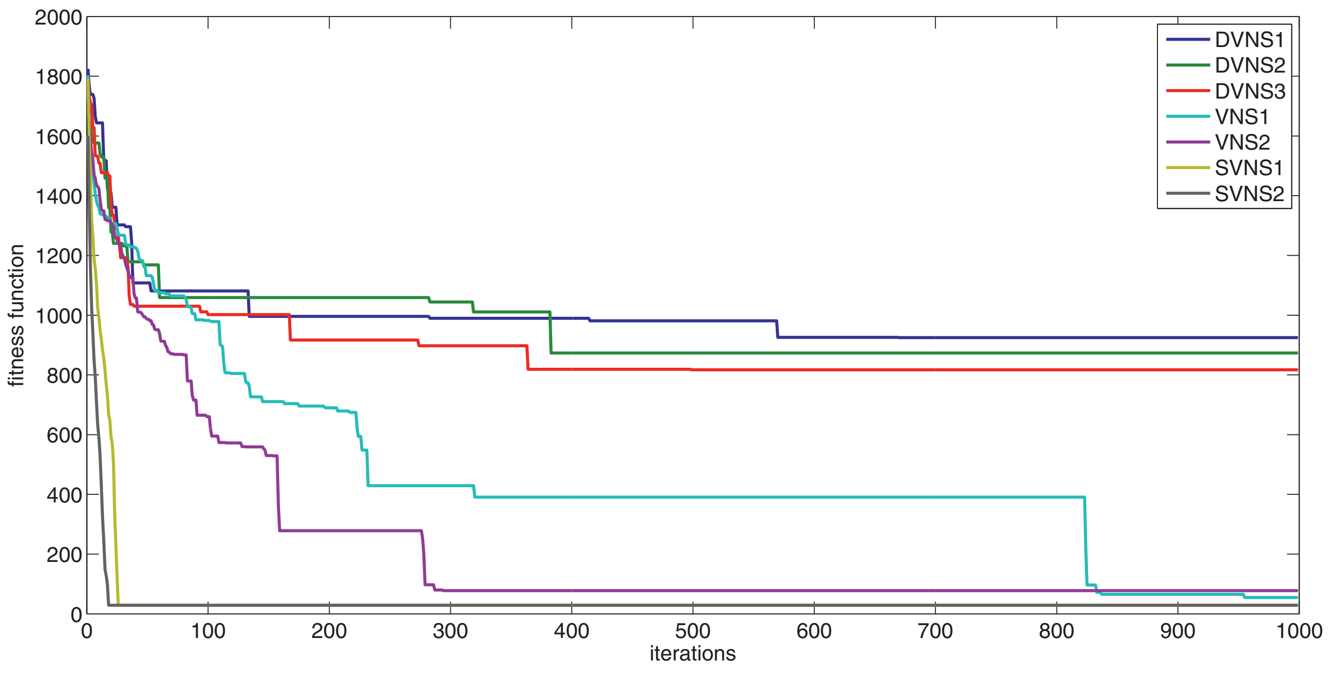

Fig. 4. Results of all VNS variants using PSOLGNT for the instance a280.

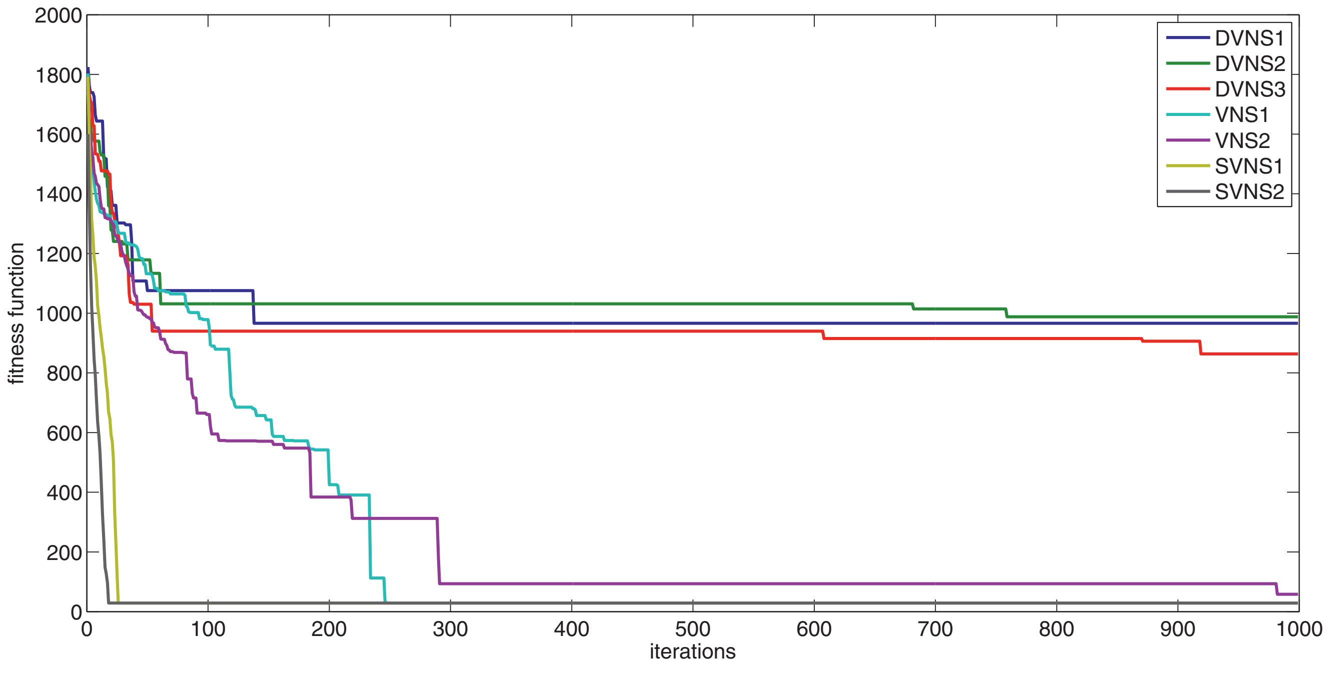

Fig. 5. Results of all VNS variants using PSOLGENT1 for the instance a280.

algorithms (except the PSOLGENT2 in one instance (eil101) without using Local Search (WLS)) find the best solution. As the number of nodes increases, the improvement of the initial solution using a VNS implementation is significant which justifies the use of the VNS algorithm for improvement of the results. In Figs. 10 and 11 the results of the three proposed versions in two specific instances using VNS1 are presented. All versions find the best solution but the algorithm that converges faster to the best solution is the PSOLGENT2. In general, we can say that if we use a continuous local search in VNS, then, the results are improved significantly. The versions of sequential VNS in general perform better than the versions of classic VNS.

In Table 7 the results of a number of metaheuristic and evolutionary algorithms are also presented. More precisely, the algorithms that are used in the comparisons are one version of a Genetic Algorithm (GA), two versions of the Differential Evolution algorithm (DE1 and DE2), a Simulated Annealing algorithm (SA), a Tabu Search algorithm (TS), two versions of a population algorithm called RSVNS1 and RSVNS2 which are algorithms that are based in the two most effective versions of VNS described in this paper but using a population that was created at random, an Ant Colony Optimization (ACO) algorithm and, finally, a Greedy Randomized Adaptive Search Procedure (GRASP). In all algorithms we use the same function evaluations as in the proposed method and the best

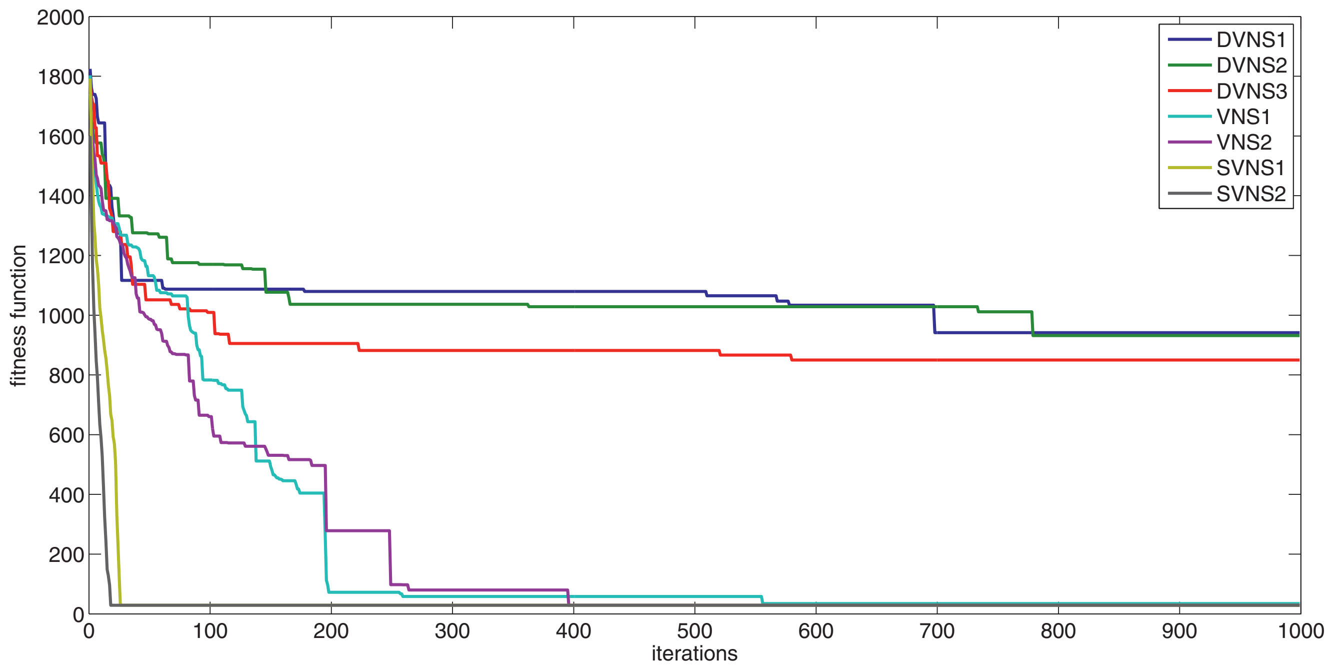

Fig. 6. Results of all VNS variants using PSOLGENT2 for the instance a280.

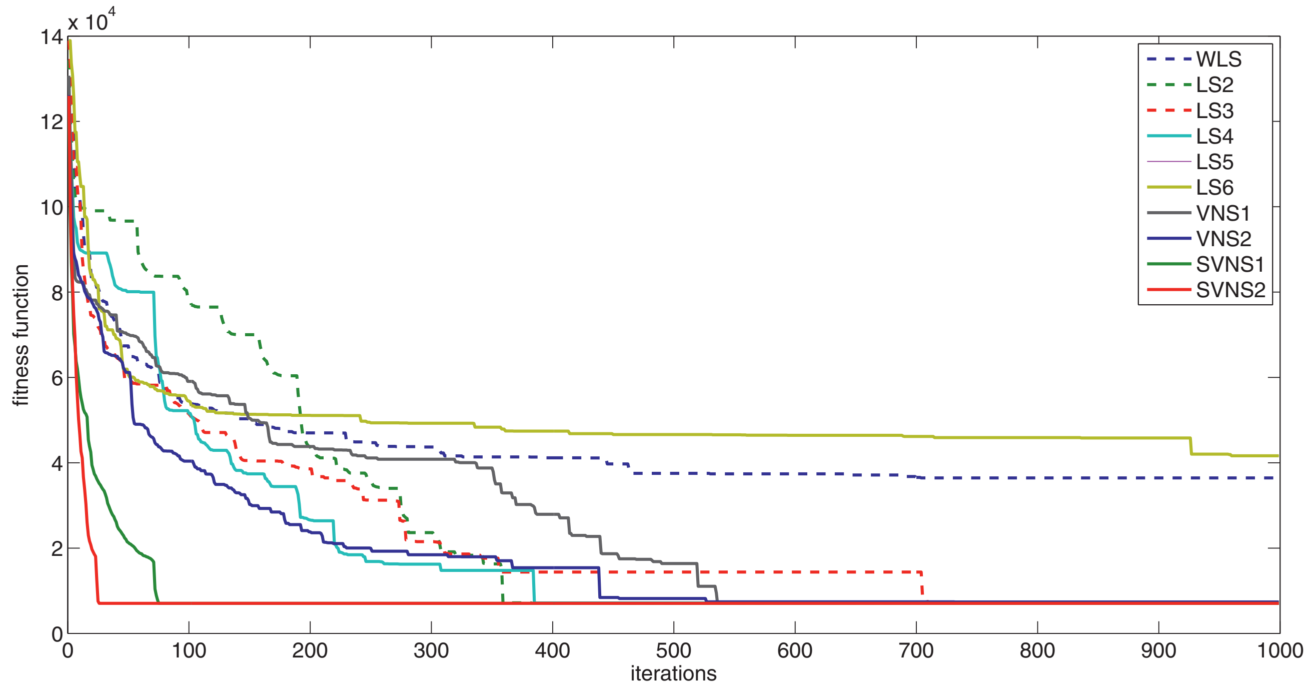

Fig. 7. Results of the PSOLGNT using all continuous local search and VNS algorithms for the instance pr439.

Table 6
Computational results for the instance pr439.

	WLS	LS1	LS2	LS3	LS4	LS5
PSOLGNT	36445.25	17954.16	7052.208	7050.47	7135.716	41630.66
PSOLGENT1	44405.02	18733.63	8690.199	17210.4	7144.314	46120.25
PSOLGENT2	43195.92	18319.13	7050.47	7052.208	7050.47	40988.59
	LS6	VNS1	VNS2	SVNS1	SVNS2	
PSOLGNT	41630.66	7050.47	7376.091	7050.47	7050.47	
PSOLGENT1	46120.25	7175.041	7709.106	7050.47	7050.47	
PSOLGENT2	40988.59	7052.208	7417.33	7050.47	7050.47	

performing VNS. The reason that we used 9 different algorithms in the comparisons is that we would like to see how the proposed algorithm performs compared to a number of different kinds of metaheuristic and evolutionary algorithms. Thus, the proposed algorithm is compared with:

- a classic evolutionary algorithm, the Genetic Algorithm (GA1) (Goldberg, 1989),
- two versions of Differential Evolution (Engelbrecht, 2007; Price, Storn, & Lampinen, 2005) algorithm (DE1 and DE2). The Differential Evolution algorithm is a competitive algorithm compared

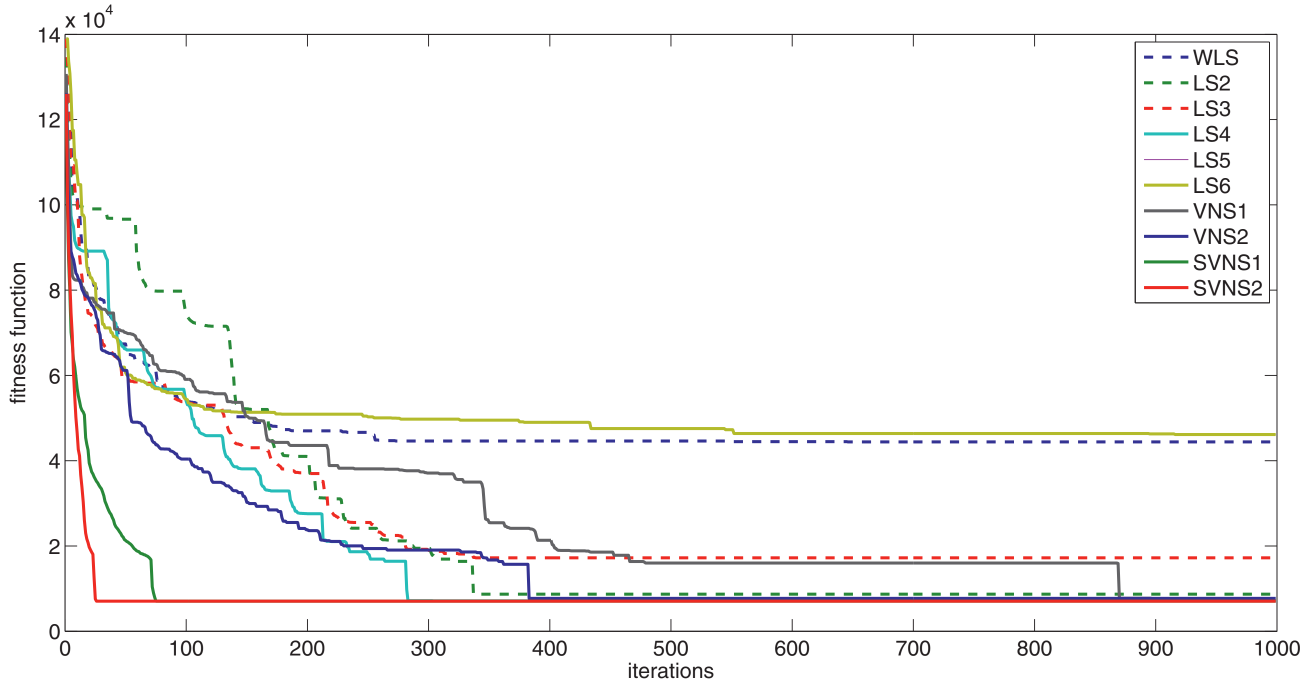

Fig. 8. Results of the PSOLGENT1 using all continuous local search and VNS algorithms for the instance pr439.

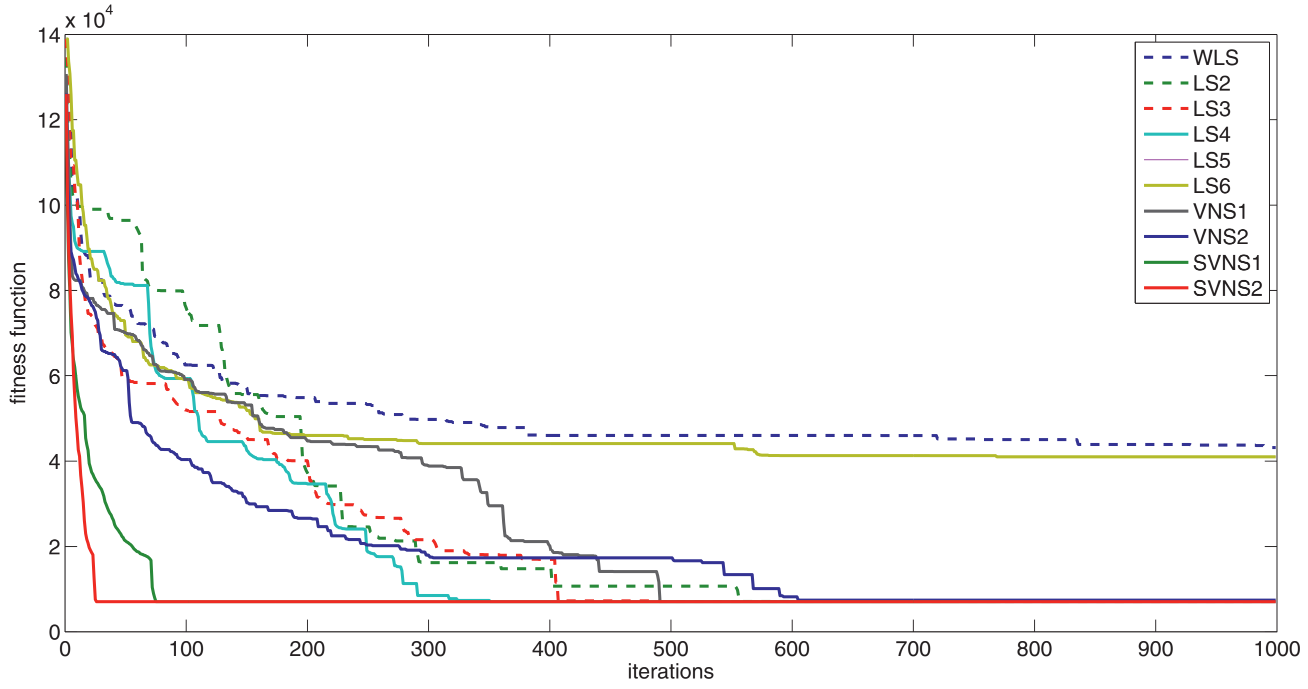

Fig. 9. Results of the PSOLGENT2 using all continuous local search and VNS algorithms for the instance pr439.

to PSO when both are applied in continuous optimization problems. The differences between the two variants of DE is in the calculation of the trial vector, in the mutation operator,

- an Ant Colony Optimization (Dorigo & Stutzle, 2004) algorithm which is the most effective nature inspired algorithm in combinatorial optimization problems,
- two versions of a population VNS algorithm (Hansen & Mladenović, 2001), the RSVNS1 and RSVNS2 which are two algorithms with exactly the same VNS procedures with the proposed algorithm but with different way to create the initial solutions and the connections of the solutions between the iterations (random way instead of PSO),
- two classic metaheuristic algorithms, the Simulated Annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1982) and the Tabu Search (TS) (Glover, 1989; 1990), and, finally,
- a Greedy Randomized Adaptive Search Procedure (Feo & Resende, 1995).

As we can see the proposed algorithm performs better than all the other algorithms used in the comparisons. More precisely, in

Table 7
Results of all of the SVNS versions and the algorithms without local search in all instances. Also results of the other metaheuristic and evolutionary algorithms.

	PSOLGNT			PSOLGENT1			PSOLGENT2		
	WLS	SVNS1	SVNS2	WLS	SVNS1	SVNS2	WLS	SVNS1	SVNS2
eil51	16.06	16.06	16.06	16.06	16.06	16.06	16.06	16.06	16.06
eil76	25.49	25.49	25.49	25.49	25.49	25.49	25.49	25.49	25.49
kroA100	2643.829	2643.829	2643.829	2643.829	2643.829	2643.829	2643.829	2643.829	2643.829
kroB100	1248.788	1248.788	1248.788	1248.788	1248.788	1248.788	1248.788	1248.788	1248.788
kroC100	1650.469	1650.469	1650.469	1650.469	1650.469	1650.469	1650.469	1650.469	1650.469
kroD100	2718.806	2718.806	2718.806	2718.806	2718.806	2718.806	2718.806	2718.806	2718.806
kroE100	3497.396	3497.396	3497.396	3497.396	3497.396	3497.396	3497.396	3497.396	3497.396
eil101	15.816	15.816	15.816	15.816	15.816	15.816	16.637	15.816	15.816
lin105	1742.875	1742.875	1742.875	1742.875	1742.875	1742.875	1742.875	1742.875	1742.875
pr144	15864.3	10950.35	10949.68	11313.3	10949.68	10949.68	11614.65	10950.35	10949.68
kroA150	1474.442	1384.003	1382.63	4129.457	1382.63	1382.63	1382.63	1382.63	1382.63
pr152	13719.13	13719.13	13719.13	13819.51	13719.13	13719.13	15016.4	13719.17	13719.13
kroA200	4311.05	2616.135	2616.135	3563.01	2616.135	2616.135	4356.84	2616.135	2616.135
kroB200	3218.153	3117.613	3117.613	3649.021	3117.628	3117.613	6532.166	3117.613	3117.613
ts225	53307.67	16621.79	16621.79	31657.91	16633.05	16621.79	50679.23	16624.08	16621.79
pr226	25434.05	10847.84	10847.84	26014.95	10950.35	10847.84	34896.34	10847.84	10847.84
Gil262	612.02	206.41	86.32	625.54	115.68	86.32	656.16	86.61	86.32
pr264	12610.49	7608.057	7608.057	18847.42	7608.06	7608.057	18330.25	8776.025	7608.057
A280	596.73	54.8	28.94	733.5	28.94	28.94	627.74	34.08	28.94
pr299	17122.38	3062.578	3062.578	17486.79	3062.578	3062.578	24861.02	3062.578	3062.578
pr439	36445.25	7050.47	7050.47	44405.02	7175.041	7050.47	43195.92	7052.208	7050.47
	GA1	DE1	DE2	SA	TS	RSVNS1	RSVNS2	ACO	GRASP
eil51	16.06	16.06	16.06	16.06	16.06	16.06	16.06	16.06	16.06
eil76	25.49	25.49	25.49	25.49	25.49	25.49	25.49	25.49	25.49
kroA100	2643.829	2643.829	2643.829	2643.829	2643.829	2643.829	2643.829	2643.829	2643.829
kroB100	1248.788	1248.788	1248.788	1248.788	1248.788	1248.788	1248.788	1248.788	1248.788
kroC100	1650.469	1650.469	1650.469	1650.469	1650.469	1650.469	1650.469	1650.469	1650.469
kroD100	2718.806	2718.806	2718.806	2718.806	2718.806	2718.806	2718.806	2718.806	2718.806
kroE100	3497.396	3497.396	3497.396	3497.396	3497.396	3497.396	3497.396	3497.396	3497.396
eil101	15.816	15.816	15.816	15.816	15.816	15.816	15.816	15.816	15.816
lin105	1742.875	1742.875	1742.875	1742.875	1742.875	1742.875	1742.875	1742.875	1742.875
pr144	11051.46	10950.35	10965.18	11058.75	10950.48	10987.21	11221.24	11217.15	12221.18
kroA150	1386.18	1385.22	1383.18	1383.47	1382.81	1391.14	1388.07	1387.15	1388.11
pr152	13725.54	13724.18	13722.24	13724.22	13720.18	13724.12	13722.17	13721.79	13722.15
kroA200	2722.35	2658.21	2659.34	2652.14	2649.24	2657.84	2671.71	2635.22	2641.78
kroB200	3358.54	3251.51	3270.18	3258.25	3179.24	3321.28	3319.31	3164.15	3189.28
ts225	17524.41	16832.17	16849.52	16749.54	16635.28	16817.21	16828.17	16671.54	16699.74
pr226	11542.35	11021.41	11147.86	10981.74	10917.43	11021.35	11081.74	10942.46	11008.41
Gil262	95.85	93.54	94.12	88.97	87.15	91.21	95.16	90.28	92.13
pr264	9214.21	7946.51	7854.15	7812.16	7691.71	7841.23	7805.14	7715.24	7798.21
A280	35.47	32.41	33.15	30.58	29.18	35.19	34.21	30.71	35.12
pr299	3174.54	3112.47	3154.21	3062.578	3062.578	3097.45	3112.28	3062.578	3115.31
pr439	7254.21	7071.31	7087.47	7061.34	7059.18	7128.18	7179.78	7054.14	7059.19

the small examples all the algorithms find the same local optimum solutions. However, as the size of the instances increases, all algorithms give inferior results than the results of the proposed algorithm. The two most effective algorithms, where their results are very close to the results of the proposed algorithm, are the Tabu Search algorithm and the Ant Colony Optimization algorithm. Then, if we made a ranking based on the effectiveness of each algorithm, the Simulated Annealing and the GRASP algorithms are in the fourth and the fifth places, respectively. After all the other are ranked with the following order, sixth is the DE1 algorithm, seventh the DE2, eighth the RSVNS1, ninth the RSVNS2 and tenth is ranked the GA1 algorithm.

6. Conclusions and future work

In this paper, a new hybridized version of Particle Swarm Optimization algorithm with Variable Neighborhood Search was presented for solving a significant Combinatorial Optimization problem, the Constrained Shortest Path problem. In the proposed algorithm, the Particle Swarm Optimization with combined Local and Global Expanding Neighborhood Topology (PSOLGENT), we give a different equation for the velocities where a fourth term has been added in the classic equation which represents the local neighbors. Thus, in the new equation the particle, except from a movement towards a new direction, a movement towards his previous best and a movement towards the global best of the swarm, could move towards the local best of its neighborhood. Also, a novel expanding neighborhood topology is used. In the proposed algorithm the neighborhood is expanding based on the quality of the solutions. We test two different alternatives. In the first one, the neighborhood is expanding when the optimal particle is not improved for a specific number of iterations. In the second alternative, each particle has its own neighborhood. Thus, while all particles begin with the same neighborhood topology when a solution of a particle is not improved for a consecutive number of iterations, then, only its neighborhood is expanding. The VNS algorithm is applied in order to optimize the particles' position. Another issue in the application of the VNS algorithm in the Constrained Shortest Path problem is which local search algorithms are suitable from this problem. Thus, in this paper a number of continuous local search algorithms are used. The algorithm is tested in a number of modified instances from the TSPLIB. We compare the algorithm with the results of two classic variants of the Particle Swarm Optimization algorithm, the inertia PSO and the con-

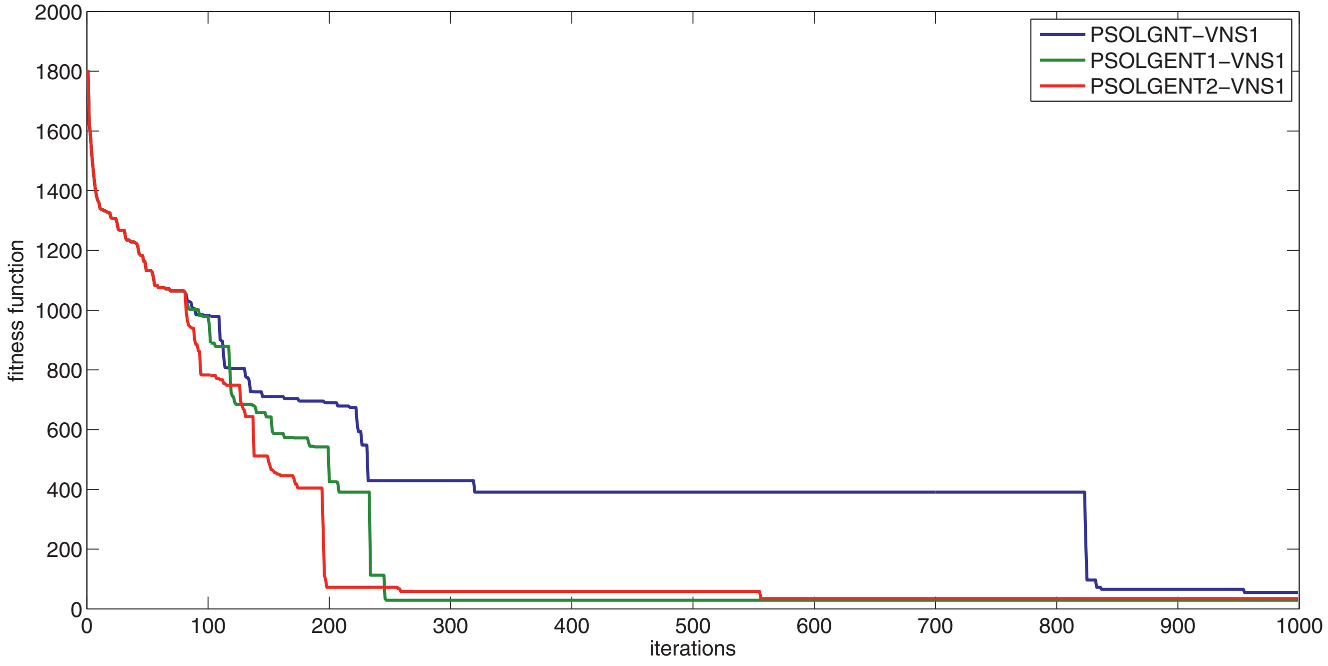

Fig. 10. Results of the three proposed algorithm for the instance a280.

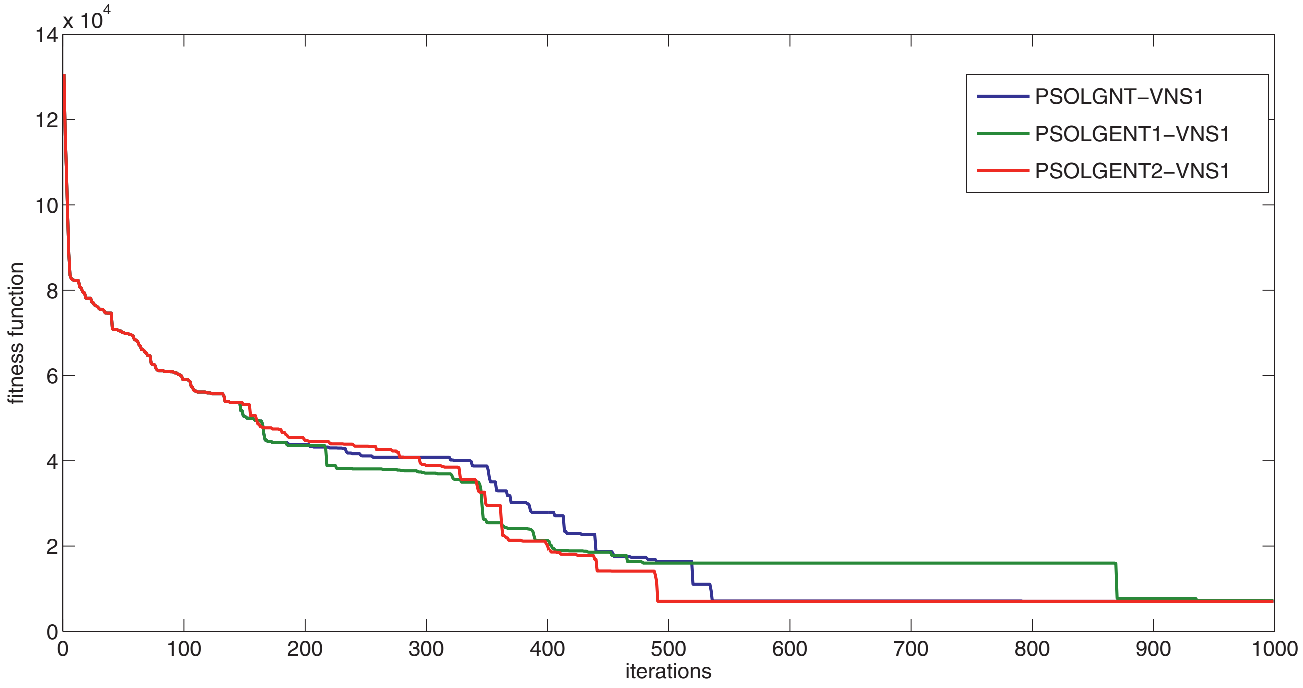

Fig. 11. Results of the three proposed algorithm for the instance pr439.

striction PSO. Also, we compare the two versions of the proposed algorithm (the one with the expanding procedure based on the best particle and the other with the expanding procedure based on each particle separately) with a variant of the algorithm with the new equation but without the expanding procedure. As a general conclusion of the computational results is that the VNS algorithm can be hybridized with a PSO algorithm in order to solve a Constrained Shortest Path problem effectively. Future research will be focused in the application of this methodology in more difficult problems.

References

Avella, P., Boccia, M., & Sforza, A. (2002). A penalty function heuristic for the resource constrained shortest path problem. *European Journal of Operational Research, 142*, 221–230.

Banks, A., Vincent, J., & Anyakoha, C. (2007). A review of particle swarm optimization. part i: Background and development. *Natural Computing, 6*(4), 467–484.

Banks, A., Vincent, J., & Anyakoha, C. (2008). A review of particle swarm optimization. part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing, 7*, 109–124.

Beasley, J. E., & Christofides, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks, 19*, 379–394.

Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing, 11*(6), 4135–4151.

Bonyadi, M. R., Li, X., & Michalewicz, Z. (2014). A hybrid particle swarm with a time-adaptive topology for constrained optimization. *Swarm and Evolutionary Computation, 18*, 22–37.

Brits, R., Engelbrecht, A. P., & Van Den Bergh, F. (2007). Locating multiple optima using particle swarm optimization. *Applied Mathematics and Computation, 189*, 1859–1883.

Cai, K., Zhang, J., Zhou, C., Cao, X., & Tang, K. (2012). Using computational intelligence for large scale air route networks design. *Applied Soft Computing, 12*(9), 2790–2800.

Carlyle, W., Royset, J. O., & Wood, R. K. (2008). Lagrangian relaxation and enumeration for solving constrained shortest-path problems. *Networks, 52*(4), 256–270.

Clerc, M. (2006). *Particle swarm optimization*. London: Hermes Science Publications, ISTE Ltd. ISBN 1-905209-04-5

Clerc, M., & Kennedy, J. (2002). The particle swarm: Explosion, stability and convergence in a multi-dimensional complex space. *IEEE Transactions on Evolutionary Computation, 6*, 58–73.

Consoli, S., Moreno-Perez, J. A., Darby-Dowman, K., & Mladenovic, N. (2008). Discrete particle swarm optimization for the minimum labelling steiner tree problem. In N. Krasnogor, G. Nicosia, M. Pavone, & D. Pelta (Eds.), *Nature inspired co-operative strategies for optimization (NICSO 2007)*. In *SCI 129* (pp. 313–322). New York.

Consoli, S., Moreno-Perez, J. A., Darby-Dowman, K., & Mladenovic, N. (2010). Discrete particle swarm optimization for the minimum labelling steiner tree problem. *Natural Computing, 9*(1), 29–46.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik, 1*, 269–271.

Dorigo, M., & Stutzle, T. (2004). *Ant colony optimization: A bradford book*. Massachusetts, London, England: The MIT Press Cambridge.

Eberhart, R. C., & Shi, Y. (2001). Particle swarm optimization: Developments, applications and resources. *IEEE Congress on Evolutionary Computation*, 81–86. doi:10.1109/CEC.2001.934374.

Engelbrecht, A. P. (2007). *Computational Intelligence: An introduction* (3rd). England: John Wiley and Sons.

Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedure. *Journal of Global Optimization, 6*, 109–133.

Formaneck, S. D., & Cozzarin, B. P. (2012). Technology adoption and training practices as a constrained shortest path problem. *Omega, 41*, 459–472.

Ghosh, S., Das, S., Kundu, D., Suresh, K., & Abraham, A. (2012). Inter-particle communication and search-dynamics of lbest particle swarm optimizers: An analysis. *Information Sciences, 182*, 156–168.

Glover, F. (1989). Tabu search I. *ORSA Journal on Computing, 1*(3), 190–206.

Glover, F. (1990). Tabu search II. *ORSA Journal on Computing, 2*(1), 4–32.

Glover, F., Laguna, M., & Marti, R. (2003). Scatter search and path relinking: Advances and applications. In F. Glover, & G. A. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 1–36). Boston: Kluwer Academic Publishers.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. INC, Massachussets: Addison-Wesley Publishing Company.

Handler, G. Y., & Zang, I. (1980). A dual algorithm for the constrained shortest path problem. *Networks, 10*, 293–310.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research, 130*, 449–467.

Hansen, P., Mladenović, N., & Moreno-Pérez, J. A. (2010). Variable neighborhood search: methods and applications. *Annals of Operations Research, 175*, 367–407.

Hu, X., & Eberhart, R. C. (2002). Multiobjective optimization using dynamic neighborhood particle swarm optimization. *Proceedings of the Congress Evolutionary Computation*, 1677–1681.

Janson, S., & Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernetics ? Part B: Cybernetics, 35*(6), 1272–1282.

Jiang, B., Wang, N., & Wang, L. (2013). Particle swarm optimization with age-group topology for multimodal functions and data clustering. *Communications in Nonlinear Science and Numerical Simulation, 18*(11), 3134–3145.

Joksch, H. (1966). The shortest route problem with constraints. *Journal of Mathematical Analysis and Applications, 14*(2), 191–197.

Kennedy, J. (1999). Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. *Proceedings of IEEE Congress on Evolutionary Computation*, 1391–1938.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks, 4*, 1942–1948.

Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. Proceedings of the 2002 IEEE congress on evolutionary computation, (pp. 1671–1676).

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1982). Optimization by simulated annealing. *Science, 220*, 671–680.

Lefebvre, M., Puget, J., & Vilim, P. (2011). Route finder: efficiently finding *k* shortest paths using constraint programming. *Principles and Practice of Constraint Programming (CP), 2011*, 42–53.

Li, X. D. (2010). Niching without niching parameters: Particle swarm optimization using a ring topology. *IEEE Transactions on Evolutionary Computation, 14*(1), 150–169.

Liang, J., Qin, A., Suganthan, P., & Baskarr, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation, 10*(3), 281–295.

Lim, W. H., & Isa, N. A. M. (2014a). Particle swarm optimization with adaptive time–varying topology connectivity. *Applied Soft Computing, 24*, 623–642.

Lim, W. H., & Isa, N. A. M. (2014b). Particle swarm optimization with increasing topology connectivity. *Engineering Applications of Artificial Intelligence, 27*, 80–102.

Liu, W., Zheng, Z., & Cai, K. Y. (2013). Bi-level programming based real-time path planning for unmanned aerial vehicles. *Knowledge-Based Systems, 44*, 34–47.

Lozano, L., & Medaglia, A. L. (2013). On an exact method for the constrained shortest path problem. *Computers and Operations Research, 40*(1), 378–384.

Marinakis, Y., & Marinaki, M. (2008). A particle swarm optimization algorithm with path relinking for the location routing problem. *Journal of Mathematical Modelling and Algorithms, 7*(1), 59–78.

Marinakis, Y., & Marinaki, M. (2010a). A hybrid genetic – Particle swarm optimization algorithm for the vehicle routing problem. *Expert Systems with Applications, 37*, 1446–1455.

Marinakis, Y., & Marinaki, M. (2010b). A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. *Computers and Operations Research, 37*, 432–442.

Marinakis, Y., & Marinaki, M. (2013a). Combinatorial neighborhood topology particle swarm optimization algorithm for the vehicle routing problem. In *Proceedings of the thirteenth European conference on evolutionary computation in combinatorial optimization (EVOCOP 2013)*. In Lecture notes in computer science*: 7832* (pp. 133–144).

Marinakis, Y., & Marinaki, M. (2013b). Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem. *Soft Computing, 17*, 1159–1173.

Marinakis, Y., Marinaki, M., & Dounias, G. (2010). A hybrid particle swarm optimization algorithm for the vehicle routing problem. *Engineering Applications of Artificial Intelligence, 23*, 463–472.

Maruta, I., Kim, T. H., Song, D., & Sugie, T. (2013). Synthesis of fixed-structure robust controllers using a constrained particle swarm optimizer with cyclic neighborhood topology. *Expert Systems with Applications, 40*(9), 3595–3605.

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation, 8*(3), 204–210.

Mohemmed, A. W., Sahoo, N. C., & Geok, T. K. (2008). Solving shortest path problem using particle swarm optimization. *Applied Soft Computing, 8*(4), 1643–1653.

Mohemmed, A. W., Sahoo, N. C., & Geok, T. K. (2010). Hybrid co-evolutionary particle swarm optimization and noising metaheuristics for the delay constrained least cost path problem. *Journal of Heuristics, 16*(4), 593–616.

Nasir, M., Das, S., Maity, D., Sengupta, S., Halder, U., & Suganthan, P. N. (2012). A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. *Information Sciences, 209*, 16–36.

Niu, B., Zhu, Y., He, X., & Wu, H. (2007). MCPSO: A multi-swarm cooperative particle swarm optimizer. *Applied Mathematics and Computation, 185*, 1050–1062.

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing, 1*(2–3), 235–306.

Peram, T., Veeramachaneni, K., & Mohan, C. K. (2003). Fitness-distance-ratio based particle swarm optimization. In *Proceedings of the 2003 IEEE swarm intelligence symposium* (pp. 174–181).

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. an overview. *Swarm Intelligence, 1*, 33–57.

Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). *Differential evolution: A practical approach to global optimization*. Berlin: Springer.

Royset, J., Carlyle, W., & Wood, R. (2009). Routing military aircraft with a constrained shortest-path algorithm. *Military Operations Research, 14*(3), 31–52.

Santos, L., Coutinho-Rodrigues, J., & Current, J. R. (2007). An improved solution algorithm for the constrained shortest path problem. *Transportation Research Part B: Methodological, 41*(7), 756–771.

Shi, Y., & Eberhart, R. C. (1998a). A modified particle swarm optimizer. Proceedings of the 1998 IEEE World Congress on Computational Intelligence, (pp. 69–73).

Shi, Y., & Eberhart, R. C. (1998b). Parameter selection in particle swarm optimization. In *Evolutionary programming VII*. In Lecture notes in computer science*: 7832* (pp. 591–600).

Sifaleras, A. (2013). Minimum cost network flows: Problems, algorithms, and software. *Yugoslav Journal of Operations Research, 23*, 3–17.

Suganthan, P. N. (1999). Particle swarm optimiser with neighborhood operator. In *Proceedings of the 1999 IEEE congress on evolutionary computation* (pp. 1958–1962).

Sura, V., & Mahadevan, S. (2011). Modeling of vertical split rim cracking in railroad wheels. *Engineering Failure Analysis, 18*(4), 1171–1183.

Tillett, T., Rao, T. M., Sahin, F., & Rao, R. (2005). Darwinian particle swarm optimization. In *Proceedings of the second Indian international conference on artificial intelligence* (pp. 1474–1487). Pune, India.

Tran, D. C., Wu, Z., & Wang, H. (2013). A novel enhanced particle swarm optimization method with diversity and neighborhood search. In *Proceedings of the 2013 IEEE international conference on systems, man, and cybernetics (SMC2013)* (pp. 180–187).

Wang, H., Lu, X., Zhang, X., Wang, Q., & Yong Deng, Y. (2014). A bio-inspired method for the constrained shortest path problem. *The Scientific World Journal, 2014*, 271280. http://dx.doi.org/10.1155/2014/271280

Wang, H., Sun, H., Li, C., Rahnamayan, S., & Pan, J. S. (2013). Diversity enhanced particle swarm optimization with neighborhood search. *Information Sciences, 223*, 119–135.

Xiao, Y., Thulasiraman, K., & Xue, G. (2006). Constrained shortest link-disjoint paths selection: A network programming based approach. *IEEE Transactions on Circuits and Systems, 53*(5), 1174–1187.

Zhang, X., Zhang, Y., Hu, Y., Deng, Y., & Mahadevan, S. (2013). An adaptive amoeba algorithm for constrained shortest paths. *Expert Systems with Applications, 40*(18), 7607–7616.