Expert Systems with Applications 37 (2010) 1446–1455

Contents lists available at ScienceDirect

Expert Systems with Applications

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

A hybrid genetic – Particle Swarm Optimization Algorithm for the vehicle routing problem

Yannis Marinakis *, Magdalene Marinaki

Technical University of Crete, Department of Production Engineering and Management, University Campus, 73100 Chania, Crete, Greece

ARTICLE INFO

Keywords:
Vehicle routing problem
Genetic algorithms
Particle Swarm Optimization

ABSTRACT

Usually in a genetic algorithm, individual solutions do not evolve during their lifetimes: they are created, evaluated, they may be selected as parents to new solutions and they are destroyed. However, research into memetic algorithms and genetic local search has shown that performance may be improved if solutions are allowed to evolve during their own lifetimes. We propose that this solution improvement phase can be assisted by knowledge stored within the parent solutions, effectively allowing parents to teach their offspring how to improve their fitness. In this paper, the evolution of each individual of the total population, which consists of the parents and the offspring, is realized with the use of a Particle Swarm Optimizer where each of them has to improve its physical movement following the basic principles of Particle Swarm Optimization until it will obtain the requirements to be selected as a parent. Thus, the knowledge of each of the parents, especially of a very fit parent, has the possibility to be transferred to its offspring and to the offspring of the whole population, and by this way the proposed algorithm has the possibility to explore more effectively the solution space. These ideas are applied in a classic combinatorial optimization problem, the vehicle routing problem, with very good results when applied to two classic benchmark sets of instances.

1. Introduction

During the last decade nature inspired intelligence became increasingly popular through the development and utilisation of intelligent paradigms in advanced information systems design. Among the most popular nature inspired approaches, when the task is optimization within complex domains of data or information, are those methods representing successful animal and micro-organism team behavior, such as swarm or flocking intelligence (birds flocks or fish schools inspired Particle Swarm Optimization (Kennedy, Eberhart, & Shi, 2001)), artificial immune systems (that mimic the biological one (Dasgupta, 1998; De Castro & Timmis, 2002)), optimized performance of bees (Baykasoglu, Ozbakor, & Tapkan, 2007), or ant colonies (ants foraging behaviors gave rise to ant colony optimization (Dorigo & Stutzle, 2004)), etc. But the most popular of the nature inspired methods are the genetic algorithms. Since their introduction (Goldberg, 1989; Holland, 1975) a very large number of applications and new ideas have been realized in the context of genetic algorithms and more general in evolutionary computation.

Usually in a genetic algorithm we have a number of discrete phases, i.e. the initialization of a population, the selection of the parents, the crossover operator, the mutation operator and the replacement of each generation. But what happens between two generations? If we would like to have a complete evolutionary algorithm we will have to observe how each individual behaves during its life. The parents try to help their offspring in order to learn and evolve and, thus, to become more competitive and to have more possibilities to survive and to become parents for the next generations. A number of different methods may be used to complete an evolutionary algorithm. One way is to observe each of the individuals, separately, without this individual having any interaction and influence with the other members of the population. In the context of the genetic algorithms, this will be realized with the use of a single or a more complex local search strategy. The other way is to have an interaction between the individuals. In this paper, this interaction is realized with the use of a Particle Swarm Optimization Algorithm (Kennedy & Eberhart, 1995). In each generation, all the individuals (parents and offspring) are considered as a single swarm and they try to improve their solution (i.e., the offspring to learn from their parents) by following the physical movement of the best particle of the whole swarm. Thus, in this paper, we focus on how each individual can be evolved with the use of the Particle Swarm

* Corresponding author. Tel.: +30 28210 37282; fax: +30 28210 69410.
E-mail addresses: marinakis@ergasya.tuc.gr (Y. Marinakis), magda@dssl.tuc.gr (M. Marinaki).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.06.085

Optimization (PSO) algorithm. At the end of each phase of the Particle Swarm Optimization the fittest of the whole swarm survive and move to the next phase of the genetic algorithm, i.e. in the selection of the parents phase of the genetic algorithm. An advantage of the application of a PSO algorithm in the evolutionary part of the algorithm is that, in contrary to other metaheuristics, there are only two variables for each individual of the population that will have to be calculated in each iteration, the position and the velocity.

The reason that a memetic algorithm (genetic algorithm with a local search phase (Moscato & Cotta, 2003)) is used instead of a classic genetic algorithm is that it is very difficult for a pure genetic algorithm to effectively explore the solution space. A combination of a global search optimization method with a local search optimization method usually improves the performance of the algorithm. In this paper instead of using a local search method to improve each individual separately, we use a global search method, Particle Swarm Optimization, and, thus, each individual does not try to improve its solution by itself but it uses knowledge from the solutions of the whole population. Another goal of our research is to achieve as good as possible results in a very short computational time. This goal led us to use two different procedures one as a speeding up technique (the Expanding Neighborhood Search – ENS) and one for the production of as good as possible initial solutions (the MPNS-GRASP). Thus, the combination of a Genetic Algorithm and a PSO algorithm with a very fast local search strategy, like the Expanding Neighborhood Search strategy (Marinakis, Migdalas, & Pardalos, 2005a, 2005b), will lead to a very fast and efficient algorithm and will reduce the computational time of the algorithm making the algorithm suitable for solving very large scale vehicle routing, and more difficult combinatorial optimization, problem. The rest of the paper is organized as follows: In the next section a description of the vehicle routing problem is presented. In the third section, the proposed algorithm, the hybrid genetic – PSO – GRASP – ENS (HybGENPSO) is presented and analyzed in detail. Computational results are presented and analyzed in the fourth section while in the last section conclusions and future research are given.

2. The vehicle routing problem

The *vehicle routing problem* (VRP) or the *capacitated vehicle routing problem (CVRP)* is often described as the problem in which vehicles based on a central depot are required to visit geographically dispersed customers in order to fulfill known customer demands. Let $G = (V, E)$ be a graph where $V = \{i_1, i_2, \ldots i_n\}$ is the vertex set (i_1 refers to the depot and the customers are indexed $i_2, \ldots, i_n$) and $E = \{(i_l, i_m) : i_l, i_m \in V\}$ is the edge set. Each customer must be assigned to exactly one of the k vehicles and the total size of deliveries for customers assigned to each vehicle must not exceed the vehicle capacity (Q_k). If the vehicles are homogeneous, the capacity for all vehicles is equal and denoted by Q. A demand q_l and a service time st_l are associated with each customer node i_l. The demand q_1 and the service time st_1 which are refered to the demand and service time of the depot are set equal to zero. The travel cost and the travel time between customers i_l and i_m is c_{lm} and tt^k_{lm}, respectively, and T_k is the maximum time allowed for a route of vehicle k. The problem is to construct a low cost, feasible set of routes – one for each vehicle. A route is a sequence of locations that a vehicle must visit along with the indication of the service it provides, where the variable x^k_{lm} is equal to 1 if the arc (i_l, i_m) is traversed by vehicle k and 0 otherwise. The vehicle must start and finish its tour at the depot. In the following we present the mathematical formulation of the VRP (Bodin, Golden, Assad, & Ball, 1983):

$$J = \min \sum_{l=1}^{n} \sum_{m=1}^{n} \sum_{k=1}^{K} c_{lm} x^k_{lm} \tag{1}$$

s.t.

$$\sum_{i_l=1}^{n} \sum_{k=1}^{K} x^k_{lm} = 1, \quad i_m = 2, \ldots, n \tag{2}$$

$$\sum_{i_m=1}^{n} \sum_{k=1}^{K} x^k_{lm} = 1, \quad i_l = 2, \ldots, n \tag{3}$$

$$\sum_{i_l=1}^{n} x^k_{lf} - \sum_{i_m=1}^{n} x^k_{fm} = 0, \quad k = 1, \ldots, K \tag{4}$$

$$i_f = 1, \ldots, n$$

$$\sum_{i_l=1}^{n} q_l \sum_{i_m=1}^{n} x^k_{lm} \leqslant Q_k, \quad k = 1, \ldots, K \tag{5}$$

$$\sum_{i_l=1}^{n} st^k_l \sum_{i_m=1}^{n} x^k_{lm} + \sum_{i_l=1}^{n} \sum_{i_m=1}^{n} tt^k_{lm} x^k_{lm} \leqslant T_k, \quad k = 1, \ldots, K \tag{6}$$

$$\sum_{i_m=2}^{n} x^k_{1m} \leqslant 1, \quad k = 1, \ldots, K \tag{7}$$

$$\sum_{i_l=2}^{n} x^k_{l1} \leqslant 1, \quad k = 1, \ldots, K \tag{8}$$

$$X \in S \tag{9}$$

$$x^k_{lm} = 0 \text{ or } 1 \quad \text{for all } i_l, i_m, k \tag{10}$$

Objective function (1) states that the total distance is to be minimized. Eqs. (2) and (3) ensure that each demand node is served by exactly one vehicle. Route continuity is represented by (4), i.e. if a vehicle enters in a demand node, it must exit from that node. Eq. (5) are the vehicle capacity constraints and (6) are the total elapsed route time constraints. Eqs. (7) and (8) guarantee that vehicle availability is not exceeded.

The vehicle routing problem was first introduced by Dantzig and Ramser (1959). As it is an NP-hard problem, a large number of approximation techniques were proposed. These techniques are classified into two main categories: Classical heuristics that were developed mostly between 1960 and 1990 (Altinkemer & Gavish, 1991; Bodin & Golden, 1981; Bodin et al., 1983; Christofides, Mingozzi, & Toth, 1979; Clarke & Wright, 1964; Desrochers & Verhoog, 1989; Fisher & Jaikumar, 1981; Foster & Ryan, 1976; Gillett & Miller, 1974; Lin, 1965; Lin & Kernighan, 1973; Mole & Jameson, 1976; Wark & Holt, 1994) and metaheuristics that were developed in the last fifteen years. Metaheuristic algorithms are classified in categories based on the used strategy. Tabu Search strategy is the most widely used technique for this problem and a number of researchers have proposed very efficient variants of the standard Tabu Search algorithm (Barbarosoglu & Ozgur, 1999; Cordeau, Gendreau, Laporte, Potvin, & Semet, 2002; Gendreau, Hertz, & Laporte, 1994; Osman, 1993; Rego, 1998; Rego, 2001; Taillard, 1993; Toth & Vigo, 2003; Xu & Kelly, 1996). Very interesting and efficient algorithms based on the concept of Adaptive Memory, according to which a set of high quality VRP solutions (elite solutions) is stored and, then, replaced from better solutions through the solution process, have been proposed (Rochat & Taillard, 1995; Tarantilis, 2005; Tarantilis & Kiranoudis, 2002). Simulated annealing (Osman, 1993), record to record travel (Golden, Wasil, Kelly, & Chao, 1998; Li, Golden, & Wasil, 2005), greedy randomized adaptive search procedure (Prins, 2008) and threshold accepting algorithms (Tarantilis, Kiranoudis, & Vassiliadis, 2002a, 2002b) are also applied efficiently in the VRP. In the last ten years a number of nature inspired metaheuristic algorithms have been applied for the solution of the vehicle routing problem. The most commonly used nature inspired methods for the solution of this problem are genetic algorithms (Baker &

Ayechew, 2003; Berger & Barkaoui, 2003; Marinakis, Migdalas, & Pardalos, 2007b; Prins, 2004), ant colony optimization (Bullnheimer, Hartl, & Strauss, 1999; Reimann, Stummer, & Doerner, 2002, 2004), honey bees mating optimization (Marinakis, Marinaki, & Dounias, 2008) and other evolutionary techniques (Cordeau, Gendreau, Hertz, Laporte, & Sormany, 2005; Mester & Braysy, 2005, 2007). The reader can find more detailed descriptions of these algorithms in the survey papers (Bodin & Golden, 1981; Bodin et al., 1983; Fisher, 1995; Gendreau, Laporte, & Potvin, 1997, 2002; Laporte, Gendreau, Potvin, & Semet, 2000; Laporte & Semet, 2002; Marinakis & Migdalas, 2002; Tarantilis, 2005) and in the books (Golden & Assad, 1988; Golden, Raghavan, & Wasil, 2008; Pereira & Tavares, 2008; Toth & Vigo, 2002).

3. Hybrid genetic – PSO – GRASP – ENS for the vehicle routing problem

3.1. General description of hybrid genetic – PSO – GRASP – ENS (HybGENPSO)

The proposed algorithm for the solution of the VRP combines a genetic algorithm, the MPNS-GRASP algorithm (Marinakis, Migdalas, & Pardalos, 2007a), the Expanding Neighborhood Search Strategy (Marinakis et al., 2005a, Marinakis, Migdalas, & Pardalos, 2005b) and a Particle Swarm Optimization Algorithm (Kennedy & Eberhart, 1995). In the following, the outline of the proposed algorithm is presented.

Initialization

(1) Create the initial population of P individuals using Multiple Phase Neighborhood Search – GRASP (MPNS-GRASP).
(2) Evaluate the fitness of each individual.
(3) Improve the fitness of each individual with the use of the Particle Swarm Optimization Strategy.

Main algorithm

(1) Set the number of generations equal to zero.
(2) **Do while** stopping criteria are not satisfied:
  2.1 **Do while** parents remain to be selected and mating:
    2.1.1 Select two parents from the current population via roulette wheel selection.
    2.1.2 Apply the crossover operator between the two parents, first cloning the common features of the two parents to the offspring and then completing the offspring using the ENS algorithm.
    2.1.3 Improve each offspring by the mutation operator (Expanding Neighborhood Search) and insert the resulting offspring to the new population.
  2.2 **Enddo**
  2.3 Improve the fitness of each individual of the new population (parents and offspring) with the use of the Particle Swarm Optimization Strategy.
  2.4 Rank the offspring and the parents via their fitness function and select for the new population a number of individuals equal to the initial population.
  2.5 Increase the generation number by one.
(3) **Enddo**
(4) Return the best individual.

3.2. Path representation

The first step in designing a genetic algorithm for a particular problem is to devise a suitable representation for the candidate solutions (Potvin, 1996). Each individual (tour) in the case of VRP, for instance, is recorded via the path representation of the tour, that is, via the specific sequence of the nodes. With this representation, there are $2n$ ways depending on which node is placed in position 1 and in which direction the tour is traversed, where n is the number of nodes, to represent the same tour depending on which node is placed in position 1. In the proposed algorithm, the node with number 1 is fixed to be always in the position 1 in the representation of every individual, overcoming, thus, the obstacle of multiple encodings of the same tour, overcoming much of the redundancy in the solution representation.

3.3. Initial population – MPNS – GRASP

Usually in a genetic algorithm there is a randomly generated initial population which may or may not necessarily contain good candidate solutions. To avoid the latter case, a modified version of the well known Greedy Randomized Adaptive Search Algorithm (GRASP), the Multiple Phase Neighborhood Search – GRASP (MPNS-GRASP) (Marinakis et al., 2007a) is used to initialize the population.

GRASP (Feo & Resende, 1995; Marinakis et al., 2005a; Resende & Ribeiro, 2003) is an iterative two phase search method which has gained considerable popularity in combinatorial optimization. Each iteration consists of two phases, a construction phase and a local search procedure. In the construction phase, a randomized greedy function is used to build up an initial solution. This randomized technique provides a feasible solution within each iteration. This solution is then exposed for improvement attempts in the local search phase. The final result is simply the best solution found over all iterations.

The construction phase can be described as a process which stepwise adds one element at a time to a partial (incomplete) solution. The choice of the next element to be added is determined by ordering all elements with respect to a greedy function, placing the best elements in a restricted candidate list (RCL), then selecting randomly from this list. The RCL may consist of the best D elements (cardinality based). Finally, the RCL is readjusted in every iteration by replacing the edge which has been included in the tour by another edge that does not belong to the RCL, namely the $(D+m)$th edge where m is the number of the current iteration. The greedy algorithm is a simple, one pass, procedure for solving the vehicle routing problem. In the second phase, a local search is initialized from these points, and the final result is simply the best solution found over all searches.

MPNS-GRASP introduces the flexibility of applying alternative greedy functions in each iteration instead of only one simple greedy function as in the classical approach. Moreover, a combination of greedy functions is also possible. The algorithm starts with one greedy function and if the results are not improving, an alternative greedy function is used instead. In these greedy functions, initially a Traveling Salesman Problem is solved (Marinakis et al., 2005a), disregarding the side constraints (capacity constraints and maximum route duration constraints) of the vehicle routing problem. Subsequently, the solution of the TSP is converted to a solution of the VRP by adding the side constraints (Bodin et al., 1983). More precisely, the first vehicle route begins from the node that corresponds to the depot and moves to the next node (customer) based on the solution of the TSP, checking if the capacity of the vehicle or if the maximum route length of the vehicle are not violated. If any of these two constraints are violated, then the vehicle returns to the depot and a new route begins.

The utilization of a simple local search in the second phase of the classical algorithm limits the chances of obtaining better solutions. Thus, MPNS-GRASP uses instead the Expanding Neighborhood Search (see Section 3.7), which is a very flexible local search strategy.

3.4. Calculation of fitness function

In VRP, the fitness of each individual is related to the route length of each tour. For each individual S its fitness is calculated by the following equation

$$fitness_S = J_{max} - J_S + 1 \tag{11}$$

where the J_{max} is the objective function value of the individual in the population with the maximum cost and J_S is the objective function value of the current individual. It should be noted that, since the probability of selecting an individual for mating is related to its fitness and since the individual with the worst cost has fitness equal to zero, it will never be selected for mating. Thus, the addition of 1 to the difference between J_{max} and J_S ensures that the worst solution is not totally excluded.

3.5. Selection probability

The selection mechanism is responsible for selecting the parent chromosome from the population and forming the mating pool. It is expected that a fitter chromosome has a higher chance of surviving on the subsequent evolution. In this work, we are using the roulette wheel selection which is one of the most common and easy to implement selection mechanisms.

3.6. Crossover operator

We propose a crossover operator which initially identifies the common characteristics of the parent individuals and, then, copies them to the offspring. This crossover operator is a kind of adaptive memory procedure. Initially, the adaptive memory has been proposed by Rochat and Taillard (1995) as a part of a tabu search metaheuristic for the solution of the vehicle routing problem. This procedure stores characteristics (tours in the vehicle routing problem) of good solutions. Each time a new good solution is found the adaptive memory is updated. In our case, in the first generation the adaptive memory is empty. In order to add a solution or a part of a solution in the adaptive memory there are a number possibilities:

(1) The candidate for the adaptive memory solution is a previous best solution and its cost function is at most 10% worst than the value of the current best solution.
(2) The candidate for the adaptive memory solution is a member of the population and its cost function is at most 10% worst than the value of the current best solution.
(3) A path is common for the best solution and for a number of individuals.

More analytically, in this crossover operator, the points are selected randomly from the adaptive memory, from the individuals and from the best solution. Thus, initially two crossover operator numbers are selected (Cr_1 and Cr_2) that control the fraction of the parameters that are selected for the adaptive memory, the individuals and the best solution. If there are common parts of the solutions then these common parts are inherited to the offspring, else the Cr_1 and Cr_2 values are compared with the output of a random number generator, $rand_i(0,1)$. If the random number is less or equal to the Cr_1 the corresponding value is inherited from the best solution, if the random number is between the Cr_1 and the Cr_2 then the corresponding value is inherited, randomly, from the one of the elite solutions that are in the adaptive memory, otherwise it is selected, randomly, from the solutions of the other individuals. Thus, if the solution of the offspring is denoted by $o_i(t)$ (t is the iteration number), the solution of the best individual is denoted by $bi_i(t)$, the solution in the adaptive memory is denoted by $ad_i(t)$ and one of the solutions of the other individuals by $p_i(t)$:

$$o_i(t) = \begin{cases} bi_i(t), & \text{if } rand_i(0,1) \leqslant Cr_1 \\ ad_i(t), & \text{if } Cr_1 < rand_i(0,1) \leqslant Cr_2 \\ p_i(t), & \text{otherwise.} \end{cases} \tag{12}$$

In each iteration the adaptive memory is updated based on the best solution. After the crossover operator, the fitness function of the offspring is calculated and if it is better than the fitness function of the parent, then, the trial vector is selected for the next generation, otherwise the parent survives for at least one more generation.

3.7. Mutation operator – Expanding Neighborhood Search

The local search method that is used in this paper is the Expanding Neighborhood Search (Marinakis et al., 2005a). Expanding Neighborhood Search (ENS) is a metaheuristic algorithm (Marinakis et al., 2005a, 2005b, 2007a, 2007b, 2008) that can be used for the solution of a number of combinatorial optimization problems with remarkable results. The main features of this algorithm are (a) the use of the Circle Restricted Local Search Moves Strategy, (b) the ability of the algorithm to change between different local search strategies and (c) the use of an expanding strategy. These features are explained in detail in the following.

In the Circle Restricted Local Search Moves – CRLSM strategy, the computational time is decreased compared to other heuristic and metaheuristic algorithms because all the edges that are not going to improve the solution are excluded from the search procedure. This happens by restricting the search space into circles around the candidate for deletion edges. It has been observed (Marinakis et al., 2005a, 2005b, 2007a), for example, in the 2-opt local search algorithm that there is only one possibility for a trial move to reduce the cost of a solution, i.e. when at least one new (candidate for inclusion) edge has cost less than the cost of one of the two old edges (candidate for deletion edges) and the other edge has cost less than the sum of the costs of the two old edges. Thus, in the Circle Restricted Local Search Moves strategy, for all selected local search strategies, circles are created around the end nodes of the candidate for deletion edges and only the nodes that are inside these circles are used in the process of finding a better solution.

In order to decrease even more the computational time and because it is more possible to find a better solution near to the endnodes of the candidate for deletion edge, we do not use from the beginning the largest possible circle but the search for a better solution begins with a circle with a small radius. For example, in the 2-opt algorithm if the length of the candidate for deletion edge is equal to A, the initial circle has radius $A/2$, then, the local search strategies are applied as they are described in the following and if the solution can not be improved inside this circle, the circle is expanding by a percentage θ(θ is determined empirically) and the procedure continues until the circle reaches the maximum possible radius which is set equal to $A + B$, where B is the length of one of the other candidate for deletion edges.

The ENS algorithm has the ability to change between different local search strategies. The idea of using a larger neighborhood to escape from a local minimum to a better one, had been proposed initially by Garfinkel and Nemhauser (1972) and recently by Hansen and Mladenovic (2001). Garfinkel and Nemhauser proposed a very simple way to use a larger neighborhood. In general, if with the use of one neighborhood a local optimum was found, then a larger neighborhood is used in an attempt to escape from the local optimum. Hansen and Mladenovic proposed a more systematical method to change between different neighborhoods, called Variable Neighborhood Search.

In the Expanding Neighborhood Search a number of local search strategies are applied inside the circle. The procedure works as follows: initially an edge of the current solution is selected (for

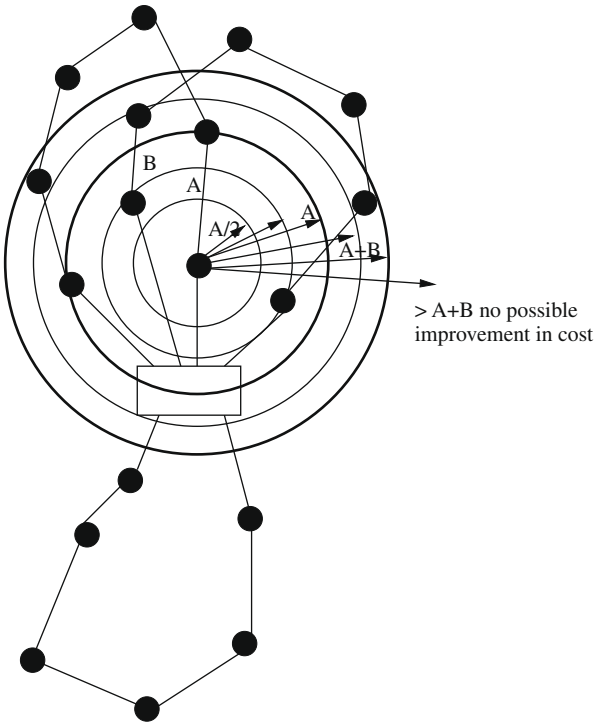

Fig. 1. Expanding neighborhood search method.

example the edge with the worst length) and the first local search strategy is applied. If with this local search strategy a better solution is not achieved, another local search strategy is selected for the same edge. This procedure is continued until a better solution is found or all local search strategies have been used. In the first case the solution is updated, a new edge is selected and the new iteration of the Expanding Neighborhood Search strategy begins, while in the second case the circle is expanded and the local search strategies are applied in the new circle until a better solution is found or the circle reach the maximum possible radius. If the maximum possible radius have been reached, then a new candidate for deletion edge is selected (Fig. 1).

The local search strategies in the Expanding Neighborhood Search for the Vehicle Routing Problem are distinguished between local search strategies for a single route and local search strategies for multiple routes. The local search strategies that are chosen and belong to the category of the single route interchange are the well known methods for the TSP, the 2-opt and the 3-opt (Lin, 1965). In the single route interchange all the routes have been created in the initial phase of the algorithm. The Local Search Strategies for Single Route Interchange try to improve the routing decisions. The Local Search Strategies for Multiple Route Interchange try to improve the assignment decisions. This, of course, increases the complexity of the algorithms but gives the possibility to improve the solution even more. The multiple route interchange strategies permit downhill and uphill moves, while the single route interchange local search strategies permit only downhill moves. The multiple route interchange local search strategies that are used are the 1–0 relocate, 2–0 relocate, 1–1 exchange, 2–2 exchange and crossing (Marinakis et al., 2007b; Marinakis et al., 2008).

3.8. Evolution of the population – Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based swarm intelligence algorithm. It was originally proposed by Kennedy and Eberhart as a simulation of the social behavior of social organisms such as bird flocking and fish schooling (Kennedy & Eberhart, 1995). PSO uses the physical movements of the individuals in the swarm and has a flexible and well-balanced mechanism to enhance and adapt to the global and local exploration abilities. Because of its easy implementation and inexpensive computation, its simplicity in coding and consistency in performance, the PSO has proved to be an effective and competitive algorithm for the optimization problem in continuous space. Most applications of PSO have concentrated on the optimization in continuous space while recently, some work has been done to the discrete optimization problem.

Since its introduction, PSO has rapidly gained popularity and proved to be a competitive and effective optimization algorithm in comparison with other metaheuristics. The PSO algorithm first randomly initializes a swarm of particles. The position of each individual (called particle) is represented by a d-dimensional vector in problem space $s_{jd}, j = 1, 2, \ldots, N$ (N is the population size), and its performance is evaluated on the predefined fitness function. Thus, each particle is randomly placed in the d-dimensional space as a candidate solution. The velocity of the j-th particle v_{jd} is defined as the change of its position. The flying direction of each particle is the dynamical interaction of individual and social flying experience. The algorithm completes the optimization through following the personal best solution of each particle and the global best value of the whole swarm. Each particle adjusts its trajectory toward its own previous best position and the previous best position attained by any particle of the swarm, namely p_{jd} and p_{gd}. In each iteration, the swarm is updated by the following equations (Kennedy & Eberhart, 1995):

$$v_{jd}(t+1) = v_{jd}(t) + c_1 rand1(p_{jd} - s_{jd}(t)) + c_2 rand2(p_{gd} - s_{jd}(t)) \tag{13}$$

$$s_{jd}(t+1) = s_{jd}(t) + v_{jd}(t+1) \tag{14}$$

where t is iteration counter; c_1 and c_2 are acceleration coefficients; $rand1$ and $rand2$ are two random numbers in [0,1]. 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, or past, target regions. Typically, these are both set to a value of 2.0, although assigning different values to c_1 and c_2 sometimes leads to improved performance (Banks, Vincent, & Anyakoha, 2007; Poli, Kennedy, & Blackwell, 2007).

The basic PSO and its variants have successfully operated for continuous optimization functions. In order to extend the application to discrete space, Kennedy and Eberhart proposed a discrete binary version of PSO (Kennedy & Eberhart, 1997) where a particle moves in a state space restricted to zero and one on each dimension where each v_{jd} represents the probability of bit s_{jd} taking the value 1. Thus, the particles trajectories are defined as the changes in the probability and v_{jd} is a measure of individuals current probability of taking 1. If the velocity is higher it is more likely to choose 1, and lower values favor choosing 0. A sigmoid function is applied to transform the velocity from real number space to probability space:

$$sig(v_{jd}) = \frac{1}{1 + exp(-v_{jd})} \tag{15}$$

In the binary version of PSO, the velocities and positions of particles are updated as the following formulas:

$$v_{jd}(t+1) = w v_{jd}(t) + c_1 rand1(p_{jd} - s_{jd}(t)) + c_2 rand2(p_{gd} - s_{jd}(t)) \tag{16}$$

$$s_{jd}(t+1) = \begin{cases} 1, & \text{if } rand3 < sig(v_{jd}) \\ 0, & \text{if } rand3 >= sig(v_{jd}) \end{cases} \tag{17}$$

where $s_{jd} \in \{0,1\}$; v_{jd} is the corresponding velocity; $sig(v_{jd})$ is calculated according to the Eq. (15), *rand*3 is a random number distributed in [0,1]. As in basic PSO, a parameter U_{max} is incorporated to limit the v_{jd} so that $sig(v_{jd})$ does not approach too closely 0 or 1 (Kennedy et al., 2001). Such implementation can ensure that the bit can transfer between 1 and 0 with a positive probability. In practice, U_{max} is often set at ± 4 (Banks et al., 2007; Poli et al., 2007). The proposed algorithm is established based on standard PSO, namely basic PSO with inertia weight developed by Shi and Eberhart in (Shi & Eberhart, 1998), where w is the inertia weight. The inertia weight controls the impact of previous histories of velocities on current velocity, which is often used as a parameter to control the trade-off between exploration and exploitation. The particle adjusts its trajectory based on information about its previous best performance and the best performance of its neighbors. The inertia weight w is also used to control the convergence behavior of the PSO. In order to reduce this weight over the iterations, allowing the algorithm to exploit some specific areas, the inertia weight w is updated according to the following equation:

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times t \tag{18}$$

where w_{max}, w_{min} are the maximum and minimum values that the inertia weight can take, and t is the current iteration (generation) of the algorithm while the $iter_{max}$ is the maximum number of iterations (generations).

One of the main problems that one has to deal with is how the particles will move from their current solution to the global optimum (optimal solution of the whole swarm) or to the local optimum (optimal solution of each particle). Usually in a discrete PSO Eq. (17) is used. The hybrid genetic PSO algorithm, instead of this formula uses a Path Relinking Strategy. Path Relinking is an intensification Strategy that is used as a way of exploring trajectories between elite solutions. Thus, the current solution of each particle is combined using this strategy either with the global or the local optimum. This approach generates new solutions by exploring trajectories that connect high-quality solutions – by starting from one of these solutions, called the *starting solution* and generating a path in the neighborhood space that leads towards the other solution, called the *target solution* (Glover, Laguna, & Marti, 2003). The roles of starting and target solutions can be interchangeable. In the first one, the worst among the two solutions plays the role of the starting solution and the other plays the role of the target solution. In the second one, the roles are changing. There is the possibility the two paths to simultaneously explored. A particle in Particle Swarm Optimization can either follow its own way, or go back to its previous optimal solution, or go towards to the global optimal solution (to the best particle in the swarm). Thus, in the hybrid genetic PSO when the particle decides to follow either the path to its previous optimal solution or the path to the global optimal solution, a path relinking strategy is applied where the current solution plays the role of the starting solution and the best particle of the swarm or the current best solution of the particle plays the role of the target solution. The trajectories between the two solutions are explored by simple swapping of two nodes of the starting solution until the starting solution becomes equal to the target solution. If in some step of the path relinking strategy a new best solution, either of the particle or of the whole swarm, is found then the current best (particle or swarm) solution is replaced with the new one and the algorithm continues.

3.9. Population replacement and termination process

As it was mentioned in the previous Section (3.8) all the individuals (parents and offspring) after crossover and mutation operators are evolved using the PSO algorithm. Afterwards the individuals who are more successful in adapting to their environment will have a better chance of surviving and reproducing, while individuals which are less fit are eliminated. Initially, all the individuals of the new population are sorted with respect to their fitness values. Subsequently, the P individuals (P equal to the number of initial population) with the best fitness functions will replace the old population. The algorithm stops when either the prespecified maximum number of generations is reached or the genetic convergence has been achieved.

4. Computational results

The whole algorithmic approach was implemented in Fortran 90 and was compiled using the Lahey f95 compiler on a Centrino Mobile Intel Pentium M 750 at 1.86 GHz, running Suse Linux 9.1. 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. Thus, the selected parameters are given in Table 1. After the selection of the final parameters, 50 different runs with the selected parameters were performed for each of the benchmark instances.

The algorithm was tested on two sets of benchmark problems. The 14 benchmark problems proposed by Christofides et al. (1979) and the 20 large scale vehicle routing problems proposed by Golden et al. (1998). Each instance of the first set contains between 51 and 200 nodes including the depot. Each problem includes capacity constraints while the problems 6–10, 13 and 14 have, also, maximum route length restrictions and non zero service times. The second set of instances contains between 200 and 483 nodes including the depot. Each problem instance includes capacity constraints while the first eight have, also, maximum route length restrictions but with zero service times. The quality of the produced solutions is given in terms of the relative deviation from the best known solution, that is $\omega = \frac{(c_{\text{HybGENPSO}} - c_{\text{BKS}})}{c_{\text{BKS}}}\%$, where $c_{\text{HybGENPSO}}$ denotes the cost of the solution found by HybGENPSO and c_{BKS} is the cost of the best known solution.

In the first column of Tables 2 and 3 the number of nodes of each instance is presented, while in the second, third and fourth columns the most important characteristics, namely the maximum capacity of the vehicles (column 2), the maximum tour length (m.t.l. – column 3) of each vehicle and the service time (s.t. – column 4) of each customer of the instances are presented. In the last six columns, the average results of the 50 runs of the algorithm (column 5), the results of the best run of the proposed algorithm (column 6), the best known solution (BKS – column 7), the average quality of the 50 runs of the algorithm (ω_{av} – column 8) the quality of the best run of the proposed algorithm (ω – column 9) and the

Table 1
Parameter values.

Parameter	Value
Number of generations for PSO	20
Number of generations for genetic algorithm	50
Population size of genetic algorithm	100
Number of swarms	1
Number of particles equal to population size	100
Probability of crossover	0.8
Probability of mutation	0.25
c_1	2
c_2	2
w_{min}	0.01
w_{max}	0.9
Size of RCL	50
θ	10%

Table 2
Results of hybrid genetic-PSO-GRASP-ENS (HybGENPSO) in the 14 Christofides benchmark instances.

Nod.	Cap.	m.t.l.	s.t.	Hyb GENPSO average	Hyb GENPSO best	BKS	ω_{av}	ω	CPU (min)
51	160	∞	0	524.61	524.61	524.61 Rochat and Taillard (1995)	0.00	0.00	0.02
76	140	∞	0	835.26	835.26	835.26 Rochat and Taillard (1995)	0.00	0.00	0.37
101	200	∞	0	826.25	826.14	826.14 Rochat and Taillard (1995)	0.01	0.00	0.41
151	200	∞	0	1029.17	1028.42	1028.42 Rochat and Taillard (1995)	0.07	0.00	1.04
200	200	∞	0	1295.38	1294.21	1291.29 Rochat and Taillard (1995)	0.32	0.23	3.01
51	160	200	10	555.43	555.43	555.43 Rochat and Taillard (1995)	0.00	0.00	0.02
76	140	160	10	909.68	909.68	909.68 Rochat and Taillard (1995)	0.00	0.00	0.34
101	200	230	10	866.76	865.94	865.94 Rochat and Taillard (1995)	0.09	0.00	1.05
151	200	200	10	1164.21	1163.41	1162.55 Rochat and Taillard (1995)	0.14	0.07	1.82
200	200	200	10	1398.27	1397.51	1395.85 Rochat and Taillard (1995)	0.17	0.12	3.65
121	200	∞	0	1043.21	1042.11	1042.11 Rochat and Taillard (1995)	0.11	0.00	0.25
101	200	∞	0	820.01	819.56	819.56 Rochat and Taillard (1995)	0.05	0.00	0.43
121	200	720	50	1545.17	1544.57	1541.14 Rochat and Taillard (1995)	0.26	0.22	0.51
101	200	1040	90	866.37	866.37	866.37 Rochat and Taillard (1995)	0.00	0.00	0.40

Table 3
Results of Hybrid Genetic-PSO-GRASP-ENS (HybGENPSO) in the 20 benchmark Golden instances.

Nod.	Cap.	m.t.l.	s.t.	Hyb GENPSO average	Hyb GENPSO best	BKS	ω_{av}	ω	CPU (min)
240	550	650	0	5673.21	5670.38	5627.54 Mester and Braysy (2007)	0.81	0.76	1.85
320	700	900	0	8466.26	8459.73	8444.50 Prins (2008)	0.26	0.18	2.21
400	900	1200	0	11112.34	11101.12	11036.22 Reimann et al. (2004)	0.69	0.59	6.35
480	1000	1600	0	13698.17	13698.17	13624.52 Prins (2004)	0.54	0.54	7.73
200	900	1800	0	6460.98	6460.98	6460.98 Tarantilis and Kiranoudis (2002)	0.00	0.00	1.28
280	900	1500	0	8473.21	8470.64	8412.8 Prins (2004)	0.72	0.69	1.49
360	900	1300	0	10218.23	10215.14	10181.75 Pisinger and Ropke (2007)	0.36	0.33	2.54
440	900	1200	0	11765.21	11750.38	11643.90 Prins (2008)	1.04	0.91	6.61
255	1000	∞	0	586.87	586.87	583.39 Mester and Braysy (2007)	0.60	0.60	1.34
323	1000	∞	0	747.18	746.56	741.56 Mester and Braysy (2007)	0.76	0.67	2.88
399	1000	∞	0	926.01	925.52	918.45 Mester and Braysy (2007)	0.82	0.77	3.44
483	1000	∞	0	1115.78	1114.31	1107.19 Mester and Braysy (2007)	0.78	0.64	8.51
252	1000	∞	0	866.38	865.19	859.11 Mester and Braysy (2007)	0.85	0.71	3.43
320	1000	∞	0	1090.23	1089.21	1081.31 Mester and Braysy (2007)	0.82	0.73	2.43
396	1000	∞	0	1355.28	1355.28	1345.23 Mester and Braysy (2007)	0.75	0.75	7.78
480	1000	∞	0	1634.49	1632.21	1622.69 Mester and Braysy (2007)	0.73	0.59	9.98
240	200	∞	0	713.72	712.18	707.79 Mester and Braysy (2007)	0.84	0.62	2.68
300	200	∞	0	1007.39	1006.31	997.52 Mester and Braysy (2005)	0.99	0.88	2.71
360	200	∞	0	1375.29	1373.24	1366.86 Mester and Braysy (2007)	0.62	0.47	3.28
420	200	∞	0	1832.29	1831.17	1820.09 Mester and Braysy (2007)	0.67	0.61	5.45

CPU time of the best run of the proposed algorithm (column 10) are presented, respectively. It can be seen from Table 2, that the algorithm, in ten of the 14 instances of the first set has reached the best known solution. For the other four instances the quality of the solutions of the best run is between 0.07% and 0.23% and the average quality for the 14 instances is 0.046%. For the 20 large scale vehicle routing problems (Table 3) the algorithm has found the best known solution in one of them, for the rest the quality is between 0.26% and 1.04% and the average quality of the best run for the twenty instances is 0.60%. Also, in these Tables the computational time needed (in minutes) for finding the best solution by HybGENPSO is presented. The CPU time needed is low for the first set of instances and only for two instances (instance 5 and 10) is somehow increased but still is very efficient. In the second set of instances, the problems are more complicated and, thus, the computational time is increased but is still less than 10 min in all instances. These results denote the efficiency of the proposed algorithm. The difference in the quality of the results of the best run of the proposed method from the average quality of the 50 runs is between 0.00% and 0.11% in the Christofides benchmark instances with average difference equal to 0.04% and is between 0.00% to 0.21% in the Golden benchmark instances with average difference equal to 0.079%. In nine instances of both sets in all 50 runs the algorithm found the best known solution. In a large number of instances only in one or two runs the algorithm did not find the optimum. However, even if the best solution was not found in all runs, the solutions found were very close to the best solutions. It should be noted that we would like to present a very fast and effective algorithm and, thus, the choice of the parameters (like the number of individuals or the number of generations) was performed in such a way in order the algorithm to combine a fast convergence with as good as possible results. This issue sometimes led the algorithm not to find the optimum. If we increase the number of individuals and the number of generations the algorithm improves even more the solutions but then the algorithm finds these solutions in more computational time.

In order to prove the contribution of each of the characteristics (metaheuristics used) in the HybGENPSO, we implement each of the main phases separately and we compare their results with the results of HybGENPSO. There are two non evolutionary algorithms, the Expanding Neighborhood Search (ENS) (columns 1 and 2 in Tables 4 and 5) and the Multiple Phase Neighborhood Search-GRASP (MPNS-GRASP) (columns 3 and 4 in Tables 4 and 5) and three evolutionary algorithm, the hybrid genetic GRASP-ENS (HybGEN) (columns 5 and 6 in Tables 4 and 5), a Genetic-PSO (GEN-PSO) algorithm (columns 7 and 8 in Tables 4 and 5) and a Genetic - PSO - ENS (GEN-PSO-ENS) algorithm (columns 9 and 10 in Tables 4 and 5). The only difference between the HybGEN

Table 4
Comparison of the proposed algorithm with ENS, MPNS-GRASP, HybGEN, GEN-PSO and GEN-PSO-ENS in the 14 Christofides benchmark instances.

ENS	ω	MPNS GRASP	ω	Hyb GEN	ω	GEN PSO	ω	GEN PSO ENS	ω	Hyb GEN PSO	ω
524.61	0.00	524.61	0.00	524.61	0.00	524.61	0.00	524.61	0.00	524.61	0.00
837.56	0.27	836.39	0.13	835.26	0.00	835.26	0.00	835.26	0.00	835.26	0.00
826.14	0.00	826.14	0.00	826.14	0.00	827.28	0.14	827.10	0.12	826.14	0.00
1034.48	0.58	1032.24	0.37	1028.42	0.00	1033.37	0.48	1029.37	0.09	1028.42	0.00
1316.18	1.91	1314.25	1.78	1299.21	0.61	1321.28	2.32	1297.38	0.47	1294.21	0.23
555.43	0.00	555.43	0.00	555.43	0.00	555.43	0.00	555.43	0.00	555.43	0.00
909.68	0.00	909.68	0.00	909.68	0.00	909.68	0.00	909.68	0.00	909.68	0.00
868.27	0.26	865.94	0.00	865.94	0.00	865.94	0.00	865.94	0.00	865.94	0.00
1178.86	1.40	1175.86	1.14	1165.13	0.22	1174.12	1.00	1165.01	0.21	1163.41	0.07
1416.14	1.45	1412.11	1.16	1402.27	0.46	1399.21	0.24	1398.21	0.17	1397.51	0.12
1043.53	0.13	1042.11	0.00	1042.11	0.00	1042.11	0.00	1042.11	0.00	1042.11	0.00
824.57	0.61	821.12	0.19	819.56	0.00	822.21	0.32	820.56	0.12	819.56	0.00
1551.24	0.65	1548.53	0.47	1545.02	0.25	1545.26	0.27	1544.68	0.23	1544.57	0.22
872.14	0.66	868.62	0.25	866.37	0.00	867.21	0.10	866.58	0.02	866.37	0.00

Table 5
Comparison of the proposed algorithm with ENS, MPNS-GRASP, HybGEN, GEN-PSO and GEN-PSO-ENS in the 20 Golden instances.

ENS	ω	MPNS GRASP	ω	Hyb GEN	ω	GEN PSO	ω	GEN PSO ENS	ω	Hyb GEN PSO	ω
5740.45	2.01	5715.19	1.56	5689.58	1.10	5721.17	1.66	5699.21	1.27	5670.38	0.76
8518.21	0.87	8490.15	0.54	8459.73	0.18	8489.21	0.53	8467.23	0.27	8459.73	0.18
11185.24	1.35	11144.39	0.98	11101.12	0.59	11221.12	1.68	11123.41	0.79	11101.12	0.59
13785.28	1.18	13752.24	0.94	13698.17	0.54	13701.28	0.56	13700.23	0.56	13698.17	0.54
6485.98	0.39	6475.19	0.22	6460.98	0.00	6474.21	0.20	6471.18	0.16	6460.98	0.00
8503.35	1.08	8492.28	0.94	8470.64	0.69	8489.32	0.91	8475.21	0.74	8470.64	0.69
10296.18	1.12	10275.17	0.92	10215.14	0.33	10277.21	0.94	10235.24	0.53	10215.14	0.33
12021.18	3.24	11918.15	2.36	11878.21	2.01	11909.25	2.28	11778.21	1.15	11750.38	0.91
595.27	2.04	589.94	1.12	586.87	0.60	587.21	0.65	586.87	0.60	586.87	0.60
759.38	2.40	749.15	1.02	746.56	0.67	748.21	0.90	748.01	0.87	746.56	0.67
937.18	2.04	935.23	1.83	925.52	0.77	934.45	1.74	925.52	0.77	925.52	0.77
1151.13	3.97	1137.17	2.71	1133.28	2.36	1136.21	2.62	1118.43	1.02	1114.31	0.64
882.17	2.68	875.14	1.87	868.17	1.05	871.28	1.42	866.32	0.84	865.19	0.71
1101.12	1.83	1098.95	1.63	1094.87	1.25	1095.21	1.29	1090.34	0.84	1089.21	0.73
1382.34	2.76	1369.16	1.78	1364.28	1.42	1366.13	1.55	1358.43	0.98	1355.28	0.75
1658.21	2.19	1651.14	1.75	1644.17	1.32	1645.24	1.39	1640.23	1.08	1632.21	0.59
719.26	1.62	715.16	1.04	712.18	0.62	721.28	1.91	717.21	1.33	712.18	0.62
1025.18	2.77	1013.17	1.57	1008.19	1.07	1010.21	1.27	1007.43	0.99	1006.31	0.88
1395.16	2.07	1384.18	1.27	1378.21	0.83	1380.01	0.96	1374.25	0.54	1373.24	0.47
1848.25	1.55	1835.18	0.83	1835.17	0.83	1835.17	0.83	1832.21	0.67	1831.17	0.61

and the HybGENPSO is the use of another phase, the evolution of the population phase that is realized with the Particle Swarm Optimization Algorithm, while the difference between the GEN-PSO and the HybGENPSO is that in GEN-PSO, the MPNS-GRASP and the ENS are not used at all. Finally, the difference between the GEN-PSO-ENS and the HybGENPSO is that in GEN-PSO-ENS the MPNS-GRASP in the initialization phase is not used at all. In all implementations, the parameters were chosen in such a way that in all algorithms the same number of function evaluations to be made. The parameter that do not affect the number of the function evaluations, e.g. RCL, c_1, c_2, etc., were set equal to the parameters of HybGENPSO and the local search strategies were the same as in HybGENPSO. In order to have the same number of function evaluations in HybGEN and HybGENPSO we increase the number of individuals and the number of generations in the HybGEN.

In Tables 4 and 5, the cost and the quality of the solutions given by the algorithms are presented. As it can be observed from Tables 4 and 5 the results are improved with the use of the proposed algorithm. More precisely, the improvement in the quality of the results of the proposed method from the ENS algorithm is between 0.00% and 1.68% in the Christofides benchmark instances with average improvement equal to 0.524% and is between 0.38% to 3.32% in the Golden benchmark instances with average improvement equal to 1.35%. The improvement in the quality of the results of the proposed method from the MPNS-GRASP algorithm is between 0.00% and 1.55% in the Christofides benchmark instances with average improvement equal to 0.347% and is between 0.22% and 2.06% in the Golden benchmark instances with average improvement equal to 0.74%. The improvement in the quality of the solutions was achieved with the addition of the Particle Swarm Optimization Algorithm. The reason is that, now, the particles moved in a more fast and efficient way to their local optimum or to the global optimum solution (to the best particle in the swarm). The improvement, now, in the quality of the results of the proposed method from the HybGEN algorithm is between 0.00% and 0.39% in the Christofides benchmark instances with average improvement equal to 0.06% and is between 0.00% and 1.71% in the Golden benchmark instances with average improvement equal to 0.31%. This last issue is very important because it is proved that the addition of the evolution of the population phase before the individuals used in the next generation improves the results of the algorithm, especially in the large scale vehicle routing instances (second set of benchmark instances) which are more difficult and time consuming. The improvement, now, in the quality of the results of the proposed method from the GEN-PSO algorithm is between 0.00% and 2.10% in the Christofides benchmark instances with average improvement equal to 0.30% and is between 0.02% and 1.97% in the Golden benchmark instances with average

improvement equal to 0.66%. The improvement, now, in the quality of the results of the proposed method from the GEN-PSO-ENS algorithm is between 0.00% and 0.25% in the Christofides benchmark instances with average improvement equal to 0.057% and is between 0% and 0.71% in the Golden benchmark instances with average improvement equal to 0.20%. This last issue is very important because it is proved that each phase of the algorithm is needed in order to have as good results as possible. Also, it should be noted that the use of the algorithm without MPNS-GRASP and ENS led to an important increase of the time that the algorithm needed to perform the same function evaluations with the HybGENPSO. Thus, each phase of the proposed algorithm is very important in the overall performance of the algorithm.

It should be, also, noted that the results of the GEN-PSO and of the GEN-PSO-ENS are very different from the results of the other four algorithms. Compared to GEN-PSO, there is a continuous improvement either to computational time needed or to the quality of the solutions obtained when the five other algorithms (ENS, MPNS-GRASP, HybGEN, GEN-PSO-ENS, HybGENPSO) are used. In GEN-PSO, in some benchmark instances the quality of the solutions is close to the quality of the solutions of the proposed algorithm. However, in many other instances the solutions are inferior even from the simple MPNS-GRASP. This is, mainly, due to the fact that the initial solutions of the GEN-PSO are created randomly. This lead us to the conclusion that if we use better initial solutions, we have a faster convergence to better results and this is the reason why we used the MPNS-GRASP and ENS in the proposed algorithm. The quality of the solutions of the GEN-PSO-ENS is much improved compared to the quality of the solutions of the GEN-PSO algorithm. This is due to the fact that now the ENS algorithm is used. The results are, only, slightly inferior from the results of the proposed algorithm (HybGENPSO) as the initial solutions in GEN-PSO-ENS are created randomly.

5. Conclusion

In this paper, a hybrid algorithmic nature inspired methodology was proposed, namely the HybGENPSO algorithm, for the effective handling of the vehicle routing problem. One of the main contributions of this paper is to show that the use of an intermediate phase between the two generations, the phase of evolution of the population, will give more efficient individuals and, thus, will improve the effectiveness of the algorithm. This is achieved by the use of a nature inspired approach, the Particle Swarm Optimization. The algorithm was applied in two set of benchmark instances and gave very satisfactory results. More specifically, in the set with the classic benchmark instances proposed by Christofides, the average quality is 0.046% while in the second set of benchmark instances proposed by Golden, the average quality is 0.60%.

References

Altinkemer, K., & Gavish, B. (1991). Parallel savings based heuristics for the delivery problem. *Operations Research, 39*(3), 456–469.

Baker, B. M., & Ayechew, M. A. (2003). A genetic algorithm for the vehicle routing problem. *Computers and Operations Research, 30*(5), 787–800.

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

Barbarosoglu, G., & Ozgur, D. (1999). A tabu search algorithm for the vehicle routing problem. *Computers and Operations Research, 26*, 255–270.

Baykasoglu, A., Ozbakor, L., & Tapkan, P. (2007). Artificial bee colony algorithm and its application to generalized assignment problem. In F. T. S. Chan & M. K. Tiwari (Eds.), *Swarm intelligence, focus on ant and particle swarm optimization* (pp. 113–144). I-Tech Education and Publishing.

Berger, J., & Barkaoui, M. (2003). A hybrid genetic algorithm for the capacitated vehicle routing problem. In *Proceedings of the genetic and evolutionary computation conference* (pp. 646–656). Chicago.

Bodin, L., & Golden, B. (1981). Classification in vehicle routing and scheduling. *Networks, 11*, 97–108.

Bodin, L., Golden, B., Assad, A., & Ball, M. (1983). The state of the art in the routing and scheduling of vehicles and crews. *Computers and Operations Research, 10*, 63–212.

Bullnheimer, B., Hartl, P. F., & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research, 89*, 319–328.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, & C. Sandi (Eds.), *Combinatorial optimization*. Chichester: Wiley.

Clarke, G., & Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12*, 568–581.

Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society, 53*, 512–522.

Cordeau, J. F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J. S. (2005). New heuristics for the vehicle routing problem. In A. Langevine & D. Riopel (Eds.), *Logistics systems: Design and optimization* (pp. 279–298). Wiley and Sons.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6*(1), 80–91.

Dasgupta, D. (Ed.). (1998). *Artificial immune systems and their application*. Heidelberg: Springer.

De Castro, L. D., & Timmis, J. (2002). *Artificial immune systems: A new computational intelligence approach*. Heidelberg: Springer.

Desrochers, M., & Verhoog, T. W. (1989). A matching based savings algorithm for the vehicle routing problem. *Les Cahiers du GERAD G-89-04*. Ecole des Hautes Etudes Commerciales de Montreal.

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

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

Fisher, M. L. (1995). In M. O. Ball, T. L. Magnanti, C. L. Momma, & G. L. Nemhauser (Eds.), *Network routing. Handbooks in operations research and management science* (Vol. 8, pp. 1–33). Amsterdam: North Holland.

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks, 11*, 109–124.

Foster, B. A., & Ryan, D. M. (1976). An integer programming approach to the vehicle scheduling problem. *Operations Research, 27*, 367–384.

Garfinkel, R., & Nemhauser, G. (1972). *Integer programming*. New York: John Wiley and Sons.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science, 40*, 1276–1290.

Gendreau, M., Laporte, G., & Potvin, J. Y. (1997). Vehicle routing: Modern heuristics. In E. H. L. Aarts & J. K. Lenstra (Eds.), *Local search in combinatorial optimization* (pp. 311–336). Chichester: Wiley.

Gendreau, M., Laporte, G., & Potvin, J. Y. (2002). Metaheuristics for the capacitated VRP. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem. Monographs on discrete mathematics and applications* (pp. 129–154). Siam.

Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operations Research, 22*, 240–349.

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.

Golden, B. L., & Assad, A. A. (1988). *Vehicle routing: Methods and studies*. Amsterdam: North Holland.

Golden, B. L., Wasil, E. A., Kelly, J. P., & Chao, I. M. (1998). The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In T. G. Crainic & G. Laporte (Eds.), *Fleet management and logistics* (pp. 33–56). Boston: Kluwer Academic Publishers.

Golden, B., Raghavan, S., & Wasil, E. (2008). *The vehicle routing problem: Latest advances and new challenges*. Springer LLC.

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

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of 1995 IEEE international conference on neural networks* (Vol. 4, pp. 1942–1948).

Kennedy, J., & Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. In *Proceedings of 1997 IEEE international conference on systems, man, and cybernetics* (Vol. 5, pp. 4104–4108).

Kennedy, J., Eberhart, R., & Shi, Y. (2001). *Swarm intelligence*. San Francisco: Morgan Kaufmann Publisher.

Laporte, G., Gendreau, M., Potvin, J. Y., & Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research, 7*, 285–300.

Laporte, G., & Semet, F. (2002). Classical heuristics for the capacitated VRP. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem, monographs on discrete mathematics and applications* (pp. 109–128). Siam.

Li, F., Golden, B., & Wasil, E. (2005). Very large-scale vehicle routing: New test problems, algorithms and results. *Computers and Operations Research, 32*(5), 1165–1179.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell Systems Technical Journal, 44*, 2245–2269.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling salesman problem. *Operation Research, 21*, 498–516.

Marinakis, Y., & Migdalas, A. (2002). Heuristic solutions of vehicle routing problems in supply chain management. In P. M. Pardalos, A. Migdalas, & R. Burkard (Eds.), *Combinatorial and global optimization* (pp. 205–236). World Scientific Publishing Co.

Marinakis, Y., Migdalas, A., & Pardalos, P. M. (2005a). Expanding neighborhood GRASP for the traveling salesman problem. *Computational Optimization and Applications, 32*, 231–257.

Marinakis, Y., Migdalas, A., & Pardalos, P. M. (2005b). A hybrid genetic-GRASP algorithm using Langrangean relaxation for the traveling salesman problem. *Journal of Combinatorial Optimization, 10*, 311–326.

Marinakis, Y., Migdalas, A., & Pardalos, P. M. (2007a). Multiple phase neighborhood search GRASP based on Lagrangean relaxation and random backtracking Lin Kernighan for the traveling salesman problem. *Journal of Combinatorial Optimization*. (Available on line doi: 10.1007/s10878-007-9104-2).

Marinakis, Y., Migdalas, A., & Pardalos, P. M. (2007b). A new bilevel formulation for the vehicle routing problem and a solution method using a genetic algorithm. *Journal of Global Optimization, 38*, 555–580.

Marinakis, Y., Marinaki, M., & Dounias, G. (2008). Honey bees mating optimization algorithm for the vehicle routing problem. In N. Krasnogor, G. Nicosia, M. Pavone, & D. Pelta (Eds.), *Nature inspired cooperative strategies for optimization – NICSO 2007. Studies in computational intelligence* (Vol. 129, pp. 139–148). Berlin: Springer-Verlag.

Mester, D., & Braysy, O. (2005). Active guided evolution strategies for the large scale vehicle routing problems with time windows. *Computers and Operations Research, 32*, 1593–1614.

Mester, D., & Braysy, O. (2007). Active guided evolution strategies for large scale capacitated vehicle routing problems. *Computers and Operations Research, 34*, 2964–2975.

Mole, R. H., & Jameson, S. R. (1976). A sequential route-building algorithm employing a generalized savings criterion. *Operational Research Quarterly, 27*, 503–511.

Moscato, P., & Cotta, C. (2003). A gentle introduction to memetic algorithms. In F. Glover & G. A. Kochenberger (Eds.), *Handbooks of metaheuristics* (pp. 105–144). Dordrecht: Kluwer Academic Publishers.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for combinatorial optimization problems. *Annals of Operations Research, 41*, 421–451.

Pereira, F. B., & Tavares, J. (2008). *Bio-inspired algorithms for the vehicle routing problem. Studies in computational intelligence* (Vol. 161). Berlin, Heideberg: Springer.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research, 34*, 2403–2435.

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

Potvin, J. Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research, 63*, 339–370.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research, 31*, 1985–2002.

Prins, C. (2008). A GRASP $\times$ evolutionary local search hybrid for the vehicle routing problem. In F. B. Pereira & J. Tavares (Eds.), *Bio-inspired algorithms for the vehicle routing problem, SCI 161* (pp. 35–53). Berlin, Heideberg: Springer.

Reimann, M., Stummer, M., & Doerner, K. (2002). A savings based ant system for the vehicle routing problem. In *Proceedings of the genetic and evolutionary computation conference* (pp. 1317–1326). New York.

Reimann, M., Doerner, K., & Hartl, R. F. (2004). D-Ants: Savings based ants divide and conquer the vehicle routing problem. *Computers and Operations Research, 31*(4), 563–591.

Rego, C. (1998). A subpath ejection method for the vehicle routing problem. *Management Science, 44*, 1447–1459.

Rego, C. (2001). Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms. *Parallel Computing, 27*(3), 201–222.

Resende, M. G. C., & Ribeiro, C. C. (2003). Greedy randomized adaptive search procedures. In F. Glover & G. A. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 219–249). Boston: Kluwer Academic Publishers.

Rochat, Y., & Taillard, E. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics, 1*, 147–167.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *Proceedings of 1998 IEEE world congress on computational intelligence* (pp. 69–73).

Taillard, E. D. (1993). Parallel iterative search methods for vehicle routing problems. *Networks, 23*, 661–672.

Tarantilis, C. D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers and Operations Research, 32*(9), 2309–2327.

Tarantilis, C. D., Kiranoudis, C. T., & Vassiliadis, V. S. (2002a). A backtracking adaptive threshold accepting metaheuristic method for the Vehicle Routing Problem. *System Analysis Modeling Simulation (SAMS), 42*(5), 631–644.

Tarantilis, C. D., Kiranoudis, C. T., & Vassiliadis, V. S. (2002b). A list based threshold accepting algorithm for the capacitated vehicle routing problem. *International Journal of Computer Mathematics, 79*(5), 537–553.

Tarantilis, C. D., & Kiranoudis, C. T. (2002). BoneRoute: An adaptive memory-based method for effective fleet management. *Annals of Operations Research, 115*(1), 227–241.

Toth, P., & Vigo, D. (2002). *The vehicle routing problem. Monographs on discrete mathematics and applications*. Siam.

Toth, P., & Vigo, D. (2003). The granular tabu search (and its application to the vehicle routing problem). *INFORMS Journal on Computing, 15*(4), 333–348.

Wark, P., & Holt, J. (1994). A repeated matching heuristic for the vehicle routing problem. *Journal of the Operational Research Society, 45*, 1156–1167.

Xu, J., & Kelly, J. P. (1996). A new network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science, 30*, 379–393.