Engineering Applications of Artificial Intelligence 23 (2010) 463–472

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

A hybrid particle swarm optimization algorithm for the vehicle routing problem

Yannis Marinakis [a,*], Magdalene Marinaki [b], Georgios Dounias [c]

[a] Technical University of Crete, Department of Production Engineering and Management, Decision Support Systems Laboratory, 73100 Chania, Greece
[b] Technical University of Crete, Department of Production Engineering and Management, Industrial Systems Control Laboratory, 73100 Chania, Greece
[c] University of the Aegean, Department of Financial and Management Engineering, Management and Decision Engineering Laboratory, 31 Fostini Str., 82100 Chios, Greece

ARTICLE INFO

Article history:
Received 2 June 2008
Received in revised form
6 April 2009
Accepted 3 February 2010
Available online 12 March 2010

Keywords:
Nature inspired intelligence
Vehicle routing problem
Metaheuristics
Particle swarm optimization
Expanding neighborhood search

ABSTRACT

This paper introduces a new hybrid algorithmic nature inspired approach based on particle swarm optimization, for successfully solving one of the most popular supply chain management problems, the vehicle routing problem. The vehicle routing problem is considered one of the most well studied problems in operations research. The proposed algorithm for the solution of the vehicle routing problem, the hybrid particle swarm optimization (HybPSO), combines a particle swarm optimization (PSO) algorithm, the multiple phase neighborhood search–greedy randomized adaptive search procedure (MPNS–GRASP) algorithm, the expanding neighborhood search (ENS) strategy and a path relinking (PR) strategy. The algorithm is suitable for solving very large-scale vehicle routing problems as well as other, more difficult combinatorial optimization problems, within short computational time. It is tested on a set of benchmark instances and produced very satisfactory results. The algorithm is ranked in the fifth place among the 39 most known and effective algorithms in the literature and in the first place among all nature inspired methods that have ever been used for this set of instances.

1. Introduction

Several biological and natural processes have been influencing the methodologies in science and technology in an increasing manner in the past years. Nature inspired intelligence becomes increasingly popular through the development and utilization of intelligent paradigms in advanced information systems design. The methods contribute to technological advances driven by concepts from nature/biology including advances in structural genomics, mapping of genes to proteins and proteins to genes, modeling of complete cell structures, functional genomics, self-organization of natural systems, etc.[1] 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 behaviour, such as swarm or flocking intelligence (birds flocks or fish schools inspired particle swarm optimization Kennedy and Eberhart, 1995; Kennedy et al., 2001), artificial immune systems (that mimic the biological one Dasgupta, 1998; De Castro and Timmis, 2002), optimized performance of bees, or ant colonies (ants foraging behaviours gave rise to ant colony optimization Dorigo and Stutzle, 2004), etc. A number of nature inspired tools have been used to solve very diverse operations and supply chain management problems, like scheduling, organization of production and vehicle routing problems (Rennard, 2006).

Particle swarm optimization (PSO) is a population-based swarm intelligence algorithm that was originally proposed by Kennedy and Eberhart (1995). PSO simulates the social behaviour of social organisms by using the physical movements of the individuals in the swarm. Its mechanism enhances and adapts to the global and local exploration. Most applications of PSO have concentrated on the optimization in continuous space while some work has been done to the discrete optimization (Kennedy and Eberhart, 1997; Shi and Eberhart, 1998). Recent complete surveys for the particle swarm optimization can be found in Banks et al. (2007, 2008) and Poli et al. (2007). The particle swarm optimization (PSO) is a very popular optimization method and its wide use, mainly during the last years, is due to the number of advantages that this method has, compared to other optimization methods. Some of the key advantages are that this method does not need the calculation of derivatives, that the knowledge of good solutions is retained by all particles and that particles in the swarm share information between them. PSO is less sensitive to the nature of the objective function, can be used for stochastic objective functions and can easily escape from local minima. Concerning its implementation, PSO can easily be programmed, has few parameters to regulate

* 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), g.dounias@aegean.gr (G. Dounias).

[1] EU's FP-6 Coordination Action NISIS: Nature Inspired Smart Information Systems (FP6-2002-IST-C) (www.nisis.de)

0952-1976/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2010.02.002

and the assessment of the optimum is independent of the initial solution.

As there are not any competitive nature inspired methods based to particle swarm optimization, at least to our knowledge, for the solution of the vehicle routing problem we would like to develop such an algorithm and to test its efficiency compared to other nature inspired and classic metaheuristic algorithms. Thus, in this paper, we demonstrate how a nature inspired intelligent technique, the particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) and three metaheuristic techniques the multiple phase neighborhood search–greedy randomized adaptive search procedure (MPNS–GRASP), the expanding neighborhood search (ENS) (Marinakis et al., 2005a) and the path relinking (PR) (Glover et al., 2003) can be incorporated in a hybrid scheme, in order to give very good results in the vehicle routing problem (VRP). Particle swarm optimization algorithms, due to their simplicity in coding and their global and local exploration abilities, are usually applied, with remarkable results, in continuous optimization problems. In this paper, we focused in the way a PSO algorithm can easily and efficiently be applied in a classic combinatorial optimization problem, like the vehicle routing problem. The main advantage of the application of a PSO algorithm in the VRP is that, in contrary to others metaheuristics, there are only two variables for each member of the population that will have to be calculated in each iteration, the position and the velocity. As we would like to increase more the efficiency of the particle swarm optimization we incorporated the three previously mentioned algorithms. More precisely:

- In order to obtain more efficient initial solutions we applied the multiple phase neighborhood search–greedy randomized adaptive search procedure (MPNS–GRASP) instead of a random creation of the initial population (Marinakis et al., 2009).
- In order to improve the solutions of each particle in the swarm and to reduce the computational time of the algorithm the expanding neighborhood search strategy (Marinakis et al., 2005a, 2005b) is utilized.
- The path relinking strategy (Glover et al., 2003) is used in order to give a more efficient way to move the particles either to their local optimum or to the global optimum of the whole swarm.

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 particle swarm optimization (HybPSO) 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 $V=\{j_0,j_1,j_2,\ldots,j_n\}$ is the vertex set ($j_i=j_0$ refers to the depot and the customers are indexed $j_i=j_1,\ldots,j_n$) and $E=\{(j_l,j_{l_1}):j_l,j_{l_1}\in V\}$ 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_{j_l} and a service time st_{j_l} are associated with each customer node j_l. The travel cost between customers j_l and j_{l_1} is $cost_{j_l j_{l_1}}$. 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. The vehicle must start and finish its tour at the depot. The most important variants of the vehicle routing problem can be found in the following: Bodin et al. (1983), Golden and Assad (1988), Marinakis and Migdalas (2002), and Toth and Vigo (2002).

The vehicle routing problem was first introduced by Dantzig and Ramser (1959). As it is an NP-hard problem, the instances with a large number of customers cannot be solved in optimality within reasonable time. For this reason 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 and metaheuristics that were developed in the last fifteen years. In the 1960s and 1970s the first attempts to solve the vehicle routing problem focused on route building, route improvement and two-phase heuristics. In the route building heuristics, the arcs are selected sequentially until a feasible solution has been created. Arcs are chosen based on some minimization cost criterion. In the route improvement heuristics, starting from one feasible solution, one more efficient solution is found by an interchange of a set of arcs. The most known of these algorithms is the Clarke and Wright method (Bodin and Golden, 1981). The 2-opt, 3-opt and Lin–Kernigham heuristics are the most known heuristics that belong in the category of route improvement heuristics. In the two phase heuristics, known as cluster first and route second heuristics, the customers are first assigned in vehicles and then a route is constructed from every cluster, like Gillet–Miller algorithm (Bodin and Golden, 1981). In this category they, also, belong the methods called route first cluster second in which first a giant traveling salesman problem (TSP) tour is constructed and then this route is decomposed into feasible vehicle routes. In the 1980s a number of mathematical programming procedures are proposed for the solution of the problem. One of these procedures is the algorithm proposed by Fisher and Jaikumar (1981).

In the 1990s, a number of algorithms, known as metaheuristics, that simulate physical phenomena, were applied for the solution of the vehicle routing problem. Simulated annealing, genetic algorithms, neural nets, tabu search, ant algorithms, together with a number of hybrid techniques are the main categories of the metaheuristic procedures. These algorithms have the ability to find their way out of local optima. In simulated annealing, this is achieved by allowing the length of the tour even to increase with a certain probability. Gradually the probability allowing the objective function value to increase is lowered until no more transformations are possible (Osman, 1993). Tabu search uses a different technique to get out of local optima. The algorithm keeps a list of forbidden transformations. In this way, it may be necessary to use a transformation that deteriorates the objective function value in the next step (Gendreau et al., 1994; Osman, 1993; Taillard, 1993; Toth and Vigo, 2003; Xu and Kelly, 1996). A neural network consists of a network of elementary nodes (neurons) that are linked through weighted connections. The nodes represent computational units, which are capable of performing a simple computation, consisting of a summation of the weighted inputs. The result of the computation of a unit constitutes its output. This output is used as an input for the nodes to which it is linked through an ongoing connection. 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 in Rochat and Taillard (1995), Tarantilis (2005), and Tarantilis and Kiranoudis (2002).

In the last 10 years, a number of nature inspired metaheuristic methods have been applied for the solution of the vehicle routing problem. The most common used nature inspired methods for the solution of this problem are genetic algorithms and ant colony optimization. Genetic algorithms mimic the evolution process in nature. Their basic operation is the mating of two tours in order to form a new tour. Moreover, they use algorithmic analogs to mutation and selection (Baker and Ayechew, 2003; Berger and Barkaoui, 2003; Marinakis et al., 2007; Prins, 2004). In the ant system artificial ants searching the solution space simulate real ants searching their environment, the objective values correspond to the quality of the food sources and an adaptive memory corresponds to the pheromone trails. In addition, the artificial ants are equipped with a local search function to guide their search through the set of feasible solutions (Bullnheimer et al., 1999; Reimann et al., 2002, 2004). The reader can find more detailed descriptions of all the previously mentioned algorithms in the survey papers (Bodin and Golden, 1981; Bodin et al., 1983; Fisher, 1995; Gendreau et al., 1997, 2002; Laporte et al., 2000; Laporte and Semet, 2002; Marinakis and Migdalas, 2002; Tarantilis, 2005).

3. Hybrid particle swarm optimization for the vehicle routing problem

3.1. General description of hybrid particle swarm optimization (HybPSO)

The proposed algorithm for the solution of the vehicle routing problem, hybrid particle swarm optimization (HybPSO), combines a particle swarm optimization algorithm, the MPNS–GRASP algorithm, the expanding neighborhood search strategy and a path relinking strategy. In this algorithm, the initial population of particles is calculated by using the MPNS–GRASP algorithm (see Section 3.2). One of the key issues in designing a successful PSO for vehicle routing problem is to find a suitable mapping between vehicle routing problem solutions and particles in PSO. Each particle is recorded via the path representation of the tour, that is, via the specific sequence of the nodes. The position of each individual (called particle) is represented by a d-dimensional vector in problem space $s_i = (s_{i1}, s_{i2}, \ldots, s_{id})$, $i = 1, 2, \ldots, N$ (N is the population size), and its performance is evaluated on the predefined fitness function. Concerning the fitness function, it should be noted that in VRP, the fitness of each particle is related to the route length of each circle and since the problem that we deal with is a minimization problem, if a feasible solution has a high objective function value then it is characterized as an unpromising solution candidate.

More precisely, initially all the solutions (particles) are represented with the path representation of the tour. For example if we have a particle with five nodes a possible path representation is the following:

1 3 5 2 4

As the calculation of the velocity of each particle is performed by the Eq. (1) (see below), the above mentioned representation should be transformed appropriately. We transform each element of the solution into a floating point interval [0,1], calculate the velocities of all particles and then convert back into the integer domain using relative position indexing (Lichtblau, 2002). Thus, initially we divided each element of the solution by the vector's largest element, and for the previous example the particle becomes:

0.2 0.6 1 0.4 0.8

The velocity of the i-th particle $v_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ 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_i and p_g. 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 (1997) proposed a discrete binary version of PSO. In our implementation instead of the formula proposed in Kennedy and Eberhart (1997) a path relinking strategy (see Section 3.4) is used. Path relinking is an intensification strategy that is used as a way of exploring trajectories between elite solutions. The velocities of the particles are updated using the following formula Kennedy and Eberhart (1995, 1997) and Shi and Eberhart (1998):

$$v_i(t+1) = wv_i(t) + c_1 rand1(p_i - s_i(t)) + c_2 rand2(p_g - s_i(t)) \quad (1)$$

where t is the iteration counter; c_1 and c_2 are the acceleration coefficients, $rand1$, $rand2$ are two random numbers in [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, or past, target regions (Kennedy and Eberhart, 1995). Typically, these are both set equal to a value of 2.0, although assigning different values to c_1 and c_2 sometimes leads to improved performance. The proposed algorithm is established based on standard PSO, namely basic PSO with inertia weight developed by Shi and 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 behaviour 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:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{t_{\max}} \times t \quad (2)$$

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 $t_{\max}$ is the maximum number of iterations (generations).

After the calculation of the velocities, the elements of the velocities' vector are transformed back into the integer domain by assigning the smallest floating value to the smallest integer, the next highest floating value to the next integer and so on. Thus, if the velocities' vector of a particle is

0.37 0.42 0.17 0.58 0.28

the backward transformation gives

3 4 1 5 2

Afterwards, we use the path relinking strategy to calculate the new position of each particle. The expanding neighborhood search strategy is utilized in order to improve the solutions of each particle in the swarm (see Section 3.3) and to reduce the computational time of the algorithm. In each iteration of the algorithm the optimal solution of the whole swarm and the optimal solution of each particle are kept. A pseudocode of the proposed hybrid particle swarm optimization algorithm is presented in Table 1.

Table 1
Hybrid particle swarm optimization algorithm for VRP.

Initialization
Select the number of Swarms
Select the number of Particles for each swarm
Generate the initial population of the particles using MPNS–GRASP
Evaluate the fitness function of each particle
Apply Expanding Neighborhood Search in each particle
Keep Optimum particle of the whole swarm
Keep Optimum solution of each particle
Main Phase
Do until the maximum number of generations has not been reached:
  Calculate the velocity of each particle
  Calculate the new position of each particle with Path Relinking
  Evaluate the new fitness function of each particle
  Apply Expanding Neighborhood Search in each particle
  Update the optimum solution of each particle
  Update the optimum particle
  Update the inertia weight
Enddo
Return the best particle (the best solution).

In the following sections, an analytical presentation of the main steps of the HybPSO is given.

3.2. Initial population

Instead of using a randomly generated initial population which may or may not necessarily contain good candidate solutions a modified version of the well known greedy randomized adaptive search procedure (grasp), the multiple phase neighborhood search–GRASP (MPNS–GRASP) is used to initialize the population.

GRASP (Feo and Resende, 1995; Marinakis et al., 2005a; Resende and 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.

That is, in the first phase, a *randomized greedy technique* provides feasible solutions incorporating both greedy and random characteristics. This 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 in a candidate list with respect to a greedy function. The heuristic is adaptive because the benefits associated with every element are updated during each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a *GRASP* is characterized by randomly choosing one of the best candidates in the list but not necessarily the top candidate. 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.

In most GRASP implementations, some type of value based restricted candidate list (RCL) has been used. In such a scheme, a RCL parameter determines the level of greediness or randomness in the construction. In MPNS–GRASP, this parameter is not used and the best promising candidate edges are selected to create the RCL. Subsequently, one of them is chosen randomly to be the next candidate for inclusion to the tour. This type of RCL is called a *cardinality based RCL construction scheme*. The restricted candidate list (RCL) of all the edges of a given graph $G = (V,E)$ is created by ordering all the edges from the smallest to the largest cost using a heap data structure. From this list, the first D edges are selected in order to form the final restricted candidate list. The candidate edge for inclusion in the tour is selected randomly from the RCL using a random number generator. 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.

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 obtained better solutions as the problem is NP-hard and more sophisticated local search algorithm are needed to improve a solution. Thus, MPNS–GRASP uses instead the expanding neighborhood search (see Section 3.3), which is a very flexible local search strategy. Almost all GRASP implementations use a termination criterion based on the maximum allowed number of iterations. Consequently, the algorithm wastes time in iterations that only add small, if any, improvements. MPNS–GRASP, on the other hand, utilizes a termination criterion based on bounds obtained through Lagrangean relaxation and subgradient optimization (Marinakis et al., 2005b). That is, first, a lower bound (LBD) is calculated and an upper bound (UBD) is estimated. Then, the parameter $e = (UBD - LBD)/UBD\%$ is computed and if its value is less than a threshold value, the algorithm stops with the current solution of MPNS–GRASP (Marinakis et al., 2005b).

3.3. Expanding neighborhood search (ENS) strategy

3.3.1. General description of ENS

Expanding neighborhood search (ENS) has been proven very efficient for the solution of the traveling salesman problem (Marinakis et al., 2005a) and of the vehicle routing problem. ENS is based on a method called *circle restricted local search moves* and, in addition, it has a number of local search phases.

In the *circle restricted local search moves—CRLSM* strategy, the computational time is decreased significantly 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 into circles around the candidate for deletion edges.

In the following, a description of the circle restricted local search moves strategy for a 2-*opt trial move* is presented. In this case, there are three possibilities based on the costs of the

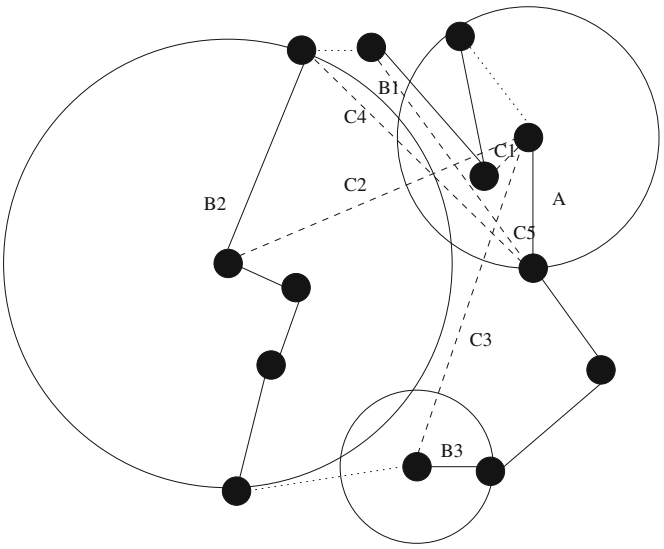

Fig. 1. Circle restricted local search moves strategy.

candidates for deletion and inclusion edges:

- If both new edges increase in cost, a 2-opt trial move cannot reduce the cost of the tour (e.g., in Fig. 1, for both new edges the costs $C2$ and $C4$ are greater than the costs $B2$ and A of both old edges).
- If one of the two new edges has cost greater than the sum of the costs of the two old edges, a 2-opt trial move, again, cannot reduce the cost of the tour (e.g. in Fig. 1, the cost of the new edge $C3$ is greater than the sum of the costs $A+B3$ of the old edges).
- The only case for which a 2-opt trial move can reduce the cost of the tour is when at least one new edge has cost less than the cost of one of the two old edges (e.g., in Fig. 1, the cost $C1$ of the new edge is less than the cost of the old edge A) and the other edge has cost less than the sum of the costs of the two old edges (e.g., $C5 < A+B1$ in Fig. 1).

Taking these observations into account, the circle restricted local search moves strategy restricts the search to edges where one of their end-nodes is inside a circle with radius length at most equal to the sum of the costs (lengths) of the two candidates for deletion edges.

The 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. On the other hand, Hansen and Mladenovic proposed a more systematical method to change between different neighborhoods, called variable neighborhood search.

On the other hand, the expanding neighborhood search method starts with one prespecified length of the radius of the circle of the CRLSM strategy. Inside this circle a number of different local search strategies are applied until all the possible trial moves have been explored and the solution cannot further be improved in this neighborhood. Subsequently, the length of the radius of the circle is increased and, again, the same procedure is repeated until the stopping criterion is activated.

The idea of searching inside a radius of the circle of the neighborhood search, the circle restricted local search moves strategy, is the most innovative feature of the ENS strategy. In expanding neighborhood search strategy, another innovative feature is the fact that the size of the neighborhood is *expanded* in each external iteration. Each different length of the neighborhood constitutes an external iteration. Initially, the size of the neighborhood, s, is defined based on the circle restricted local search moves strategy, for example $s = A/2$, where A is the cost (length) of one of the candidates for deletion edges. For the selected size of the neighborhood, a number of different local search strategies are applied until all the possible trial moves have been explored and the solution can not further be improved in this neighborhood. The local search strategies are changed based on two conditions, first if the current local search strategy finds a local optimum and second if the quality of the current solution remains greater than the threshold number b_1 for a number of internal iterations. Subsequently, the quality of the current solution is compared with the current Lagrangian lower bound. If the quality of the solution is less than the threshold number e_1 the algorithm stops, otherwise the neighborhood is expanded by increasing the length of the radius of the CRLSM strategy s by a percentage θ (e.g. $\theta = 10\%$), the Lagrangian lower bound is updated and the algorithm continues. When the length of the radius of the CRLSM strategy is equal to A, the length continues to increase until the length becomes equal to $A+B$, where B is the length of the other candidate for deletion edge. If the length of the radius of the CRLSM strategy is equal to $A + B$, and no stopping criterion has been, already, activated, then the algorithm terminates with the current solution. In Fig. 2, the expanding neighborhood search strategy is presented. A feature that has to be mentioned is that with this method if two routes are in opposite directions around the depot there is no possibility to be checked for a possible exchange between them in order to have an improvement in the cost, because the search for better solutions is restricted in neighborhood routes.

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

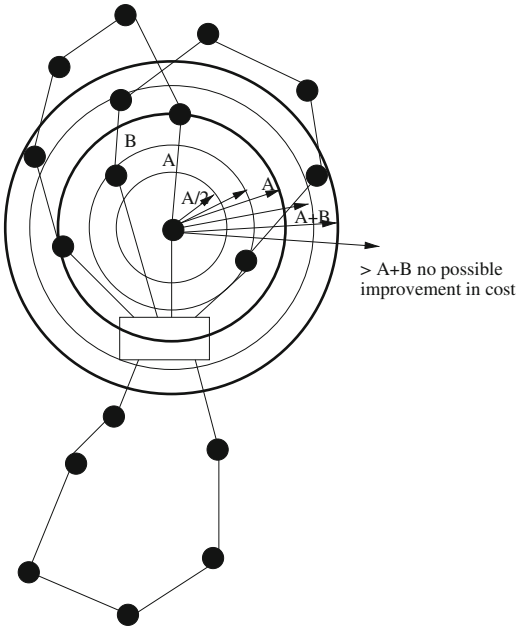

Fig. 2. Expanding neighborhood search strategy.

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 even more the solution. 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 (Gendreau et al., 1997).

Table 2
Parameter values.

Parameter	Value
Number of swarms	1
Number of particles	20
Number of generations	100
c_1	2
c_2	2
$w_{\min}$	0.01
$w_{\max}$	0.9
Size of RCL	50

3.4. Path relinking

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 et al., 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 HybPSO 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.

Table 3
Results of HybPSO in christofides benchmark instances.

	Nodes	Capacity	Max. tour length.	Service time	HybPSO	Best Known Solution	Quality (%)	CPU (min)
1	51	160	∞	0	524.61	524.61	0.00	0.05
2	76	140	∞	0	835.26	835.26	0.00	0.21
3	101	200	∞	0	826.14	826.14	0.00	0.32
4	151	200	∞	0	1029.54	1028.42	0.11	1.01
5	200	200	∞	0	1294.13	1291.45	0.22	2.15
6	51	160	200	10	555.43	555.43	0.00	0.05
7	76	140	160	10	909.68	909.68	0.00	0.28
8	101	200	230	10	868.45	865.94	0.29	0.89
9	151	200	200	10	1164.35	1162.55	0.15	1.57
10	200	200	200	10	1396.18	1395.85	0.02	3.01
11	121	200	∞	0	1044.03	1042.11	0.18	0.53
12	101	200	∞	0	819.56	819.56	0.00	0.38
13	121	200	720	50	1544.18	1541.14	0.20	0.41
14	101	200	1040	90	866.37	866.37	0.00	0.37

Table 4
Comparison of the proposed algorithm with other PSO implementations.

	PSO		PSO-MPNS–GRASP		PSO-MPNS–GRASP-ENS		HybPSO	
	Cost	CPU (min)	Cost	CPU (min)	Cost	CPU (min)	Cost	CPU (min)
1	524.61	1.07	524.61	1.02	524.61	0.08	524.61	0.05
2	845.24	1.85	839.18	1.87	838.21	0.25	835.26	0.21
3	832.91	1.65	828.45	1.68	828.21	0.31	826.14	0.32
4	1049.24	2.26	1037.11	2.17	1037.18	1.08	1029.54	1.01
5	1361.24	4.01	1312.37	4.18	1308.12	2.12	1294.13	2.15
6	555.43	1.12	555.43	0.99	555.43	0.12	555.43	0.05
7	918.27	1.97	910.21	1.87	909.68	0.23	909.68	0.28
8	889.12	2.67	879.28	2.35	875.25	0.97	868.45	0.89
9	1197.23	3.28	1172.01	3.01	1171.92	1.87	1164.35	1.57
10	1428.21	5.40	1409.19	5.87	1407.34	3.09	1396.18	3.01
11	1057.01	1.41	1045.45	1.32	1045.21	0.55	1044.03	0.53
12	827.21	1.65	819.56	1.41	819.56	0.61	819.56	0.38
13	1551.21	1.37	1545.98	1.28	1545.11	0.78	1544.18	0.41
14	876.45	1.87	866.37	1.70	866.37	0.44	866.37	0.37

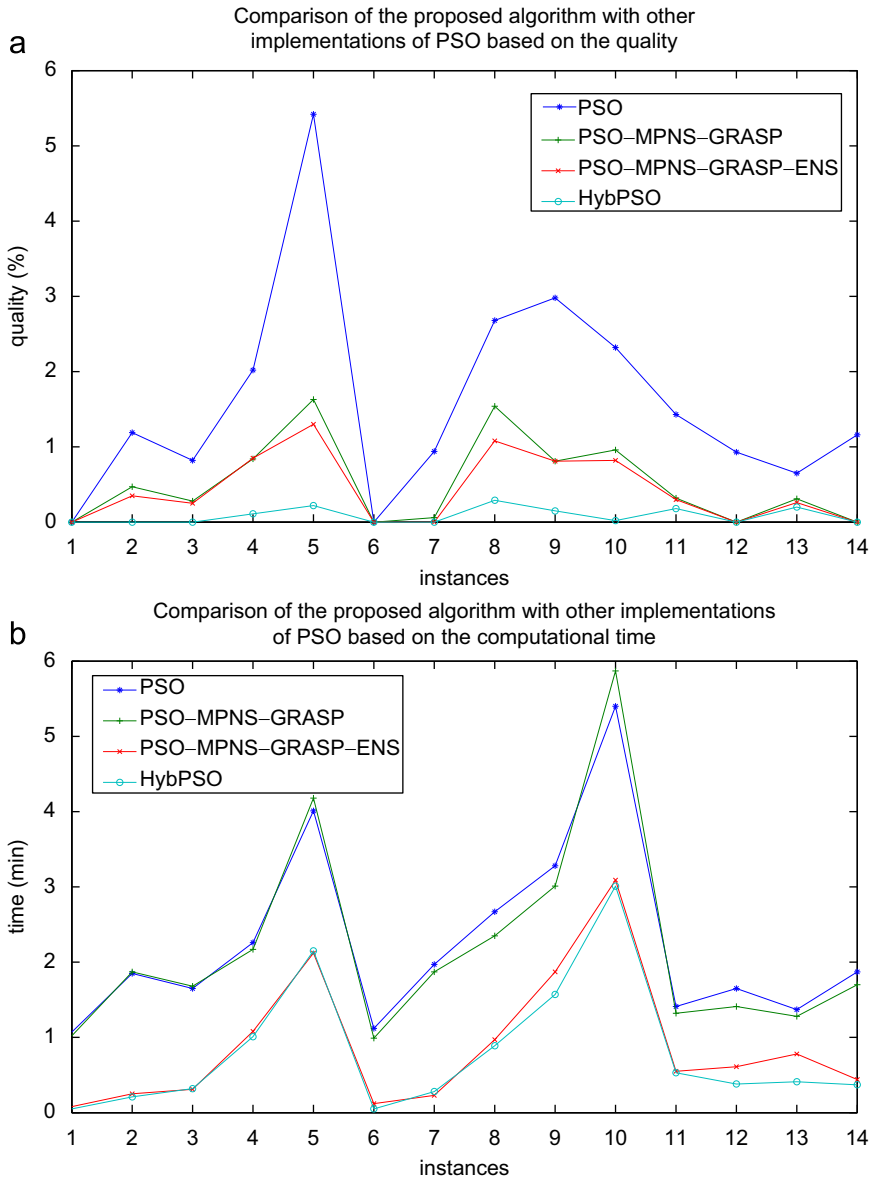

Fig. 3. Comparison of the proposed algorithm with other implementations of PSO based on the quality (a) and based on the computational time (b).

4. Computational results for the vehicle routing problem

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 2.

The algorithms were tested on a set of benchmark problems, the 14 benchmark problems proposed by Christofides et al. (1979). Each instance of the set contains between 51 and 200 nodes including the depot. The location of the nodes is defined by their Cartesian co-ordinates and the travel cost from node i to j is assumed to be the respective Euclidean distance. Each problem includes capacity constraints while the problems 6–10, 13 and 14 have, also, maximum route length restrictions and non-zero service times. For the first 10 problems, nodes are randomly located over a square, while for the remaining ones, nodes are distributed in clusters and the depot is not centered.

The efficiency of the HybPSO algorithm is measured by the quality of the produced solutions. The quality is given in terms of the relative deviation from the best known solution, that is $\omega = (c_{HybPSO} - c_{opt})/c_{opt}\%$, where c_{HybPSO} denotes the cost of the solution found by HybPSO and c_{opt} is the cost of the best known solution. It can be seen from Table 3, that the HybPSO algorithm, in half of the instances proposed by Christofides has reached the best known solution. For the rest instances proposed by Christofides the quality of the solutions is between 0.02% and 0.29% with average quality 0.084%. Also, in this Table the computational time needed (in minutes) for finding the best solution by HybPSO is presented. The CPU time needed is significantly low and only for one instance (instance 10) is somehow increased but still is very efficient. These results denote the efficiency of the proposed algorithm.

In order to give the significance of each of the characteristics (metaheuristics used) of the HybPSO, we implement a number of different versions of a particle swarm optimization algorithm for

Table 5
Comparison of various heuristic and metaheuristic algorithms with HybPSO in Christofides benchmark instances.

Rank	Algorithm	Quality (%)	Rank	Algorithm	Quality (%)
1	RT (Rochat and Taillard, 1995)	0.00	21	Wark Holt (Wark and Holt, 1994)	0.635
2	AGES best (Cordeau et al., 2005; Mester and Braysy, 2007)	0.03	22	Granular Tabu Search (GTS) (Toth and Vigo, 2003)	0.64
3	Taillard (Taillard, 1993)	0.051	23	UTSA (Cordeau et al., 2002)	0.689
4	AGES fast (Cordeau et al., 2005; Mester and Braysy, 2007)	0.07	24	St-TABUROUTE (Gendreau et al., 1994)	0.863
5	**HybPSO**	0.084	25	Osman Tabu Search (Osman, 1993)	1.011
6	Best-Prins (Prins, 2004)	0.085	26	BHS (Bullnheimer et al., 1999)	1.511
7	Best-SEPAS (Tarantilis, 2005)	0.182	27	SEC (Rego, 1998)	1.539
8	St-SEPAS (Tarantilis, 2005)	0.195	28	XuKelly (Xu and Kelly, 1996)	1.718
9	Best-TABUROUTE (Gendreau et al., 1994)	0.198	29	Osman Simulated Annealing (Osman, 1993)	2.105
10	BoneRoute (Tarantilis and Kiranoudis, 2002)	0.226	30	2-Petal (Foster and Ryan, 1976)	2.430
11	Stand-Prins (Prins, 2004)	0.235	31	Fisher–Jaikumar (Fisher and Jaikumar, 1981)	2.659
12	RSD (Reimann et al., 2002)	0.383	32	B-SL (Laporte and Semet, 2002)	3.295
13	VRPBilevel (Marinakis et al., 2007)	0.479	33	Gavish (Altinkemer and Gavish, 1991)	3.415
14	D-Ants (Reimann et al., 2004)	0.481	34	Christofides–Mignozzi–Toth (Christofides et al., 1979)	5.055
15	Stand-HGA (Berger and Barkaoui, 2003)	0.485	35	1-Petal (Foster and Ryan, 1976)	5.937
16	LBTA (Tarantilis et al., 2002b)	0.498	36	CW (Clarke and Wright, 1964)	6.724
17	BAGA (Baker and Ayechew, 2003)	0.504	37	Desrochers (Desrochers and Verhoog, 1989)	6.922
18	BATA (Tarantilis et al., 2002a)	0.525	38	Sweep (Gillett and Miller, 1974)	7.177
19	PTC (Rego, 2001)	0.549	39	Mole–Jameson (Mole and Jameson, 1976)	10.906
20	Barbarosoglu (Barbarosoglu and Ozgur, 1999)	0.57			

Table 6
Comparison of other nature inspired algorithms with HybPSO in Christofides benchmark instances.

Rank	Algorithm	Quality (%)
1	**HybPSO**	0.084
2	Best-Prins (Prins, 2004)	0.085
3	Stand-Prins (Prins, 2004)	0.235
4	RSD (Reimann et al., 2002)	0.383
5	VRPBilevel (Marinakis et al., 2007)	0.479
6	D-Ants (Reimann et al., 2004)	0.481
7	Stand-HGA (Berger and Barkaoui, 2003)	0.485
8	BAGA (Baker and Ayechew, 2003)	0.504
9	BHS (Bullnheimer et al., 1999)	1.511

VRP. In these implementations the basic characteristics of the HybPSO are not included in order to prove the contribution of each of the characteristics of the proposed algorithm. More precisely, initially a particle swarm optimization algorithm is tested without the MPNS–GRASP, the expanding neighborhood search strategy and the path relinking strategy (columns 2 and 3 of Table 4), afterwards the MPNS–GRASP strategy (columns 4 and 5 of Table 4), and the ENS strategy (columns 6 and 7 of Table 4) are added , finally in the last two columns of the Table the results of the HybPSO are presented since the path relinking strategy is added. In the implementations where the path relinking strategy is not included the equation proposed by Kennedy and Eberhart (1997) is used in order the particle to decide if it will follows 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). In all implementations, the parameters were set equal to the parameters of HybPSO and the local search strategies were the same as in HybPSO. In Table 4 and Fig. 3 the cost and the computational time of all implementations are presented. From this Table and this figure it can be observed that the use of each of the characteristics in the HybPSO improves significantly either the quality of the solution or the computational time or both of them. More precisely, the addition of the MPNS–GRASP in the initial PSO algorithm gave much better results regarding the quality of the solution but the computational time was not improved at all. The computational time was improved significantly with the addition of the expanding neighborhood search strategy, but the results were almost the same as in the previous case. The significant improvement in the quality of the solutions was achieved with the addition of the path relinking strategy. The reason is that, now, the particles moved in more fast and efficient way to their local optimum or to the global optimum solution (to the best particle in the swarm). It can, also, be observed that in some instances namely for instances 1 and 6, all four implementations found the same solutions but the significant difference is the time that they needed to find this solution. More precisely, while in the first implementation the computational time to find the optimum is equal to 1.07 min and in the second implementation is equal to 1.05 min, the addition of the ENS reduce the computational time (third implementation) to 0.08 min and the addition of the path relinking (HybPSO) algorithm reduced even more the computational time to find the optimum to 0.05 min.

The results obtained by the proposed algorithm are also compared to the results of the most efficient algorithms that have ever been presented for the vehicle routing problem. The VRP belongs to the class of NP-hard optimization problems. This means that no polynomial time algorithm is known for its solution. Algorithms for solving the VRP may be divided in two classes, the exact algorithms and the approximation algorithms. The approximation algorithms for the VRP are classified into two main categories, heuristics and metaheuristics algorithms. From these two categories, the most known and the most efficient were chosen for the comparisons. The most classical heuristics algorithms are not competitive with the metaheuristic algorithms but they are included in the comparisons for reasons of completeness.

The heuristic algorithms that are used in the comparisons with the proposed algorithm are the Clarke and Wright (1964) algorithm, the 1-petal and the 2-petal algorithms (Foster and Ryan, 1976), the Fisher and Jaikumar (1981) algorithm, the Wark and Holt (1994) algorithm, the B-SL Gavish algorithm (Altinkemer and Gavish, 1991), the Christofides et al. (1979) algorithm, the Desrochers and Verhoog (1989) algorithm , the sweep algorithm (Gillett and Miller, 1974) and, finally, the Mole and Jameson (1976) algorithm.

Metaheuristic algorithms are classified in categories based on the used strategy. In these comparisons, algorithms that are based on Neural Networks are not included, because their results are not competitive with the other metaheuristic algorithms. 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 (Taillard, TABUROUTE, Osman Tabu

Table 7
Running times of various heuristic and metaheuristic algorithms in Christofides benchmark instances.

Algorithm	CPU (min)	Computer used
AGES fast (Cordeau et al., 2005; Mester and Braysy, 2007)	0.27	Pentium IV 2 GHz
HybPSO	0.80	Centrino Mobile Intel Pentium M 750 at 1.86 GHz
SEC (Rego, 1998)	2.32	HP 9000/712
D-Ants (Reimann et al., 2004)	3.28	Pentium 900 MHz
GTS (Toth and Vigo, 2003)	3.84	Pentium 200 MHz
Best-Prins (Prins, 2004)	5.2	Pentium 1000 MHz
Stand-Prins (Prins, 2004)	5.2	Pentium 1000 MHz
St-SEPAS (Tarantilis, 2005)	5.6	Pentium II 400 MHz
BoneRoute (Tarantilis and Kiranoudis, 2002)	6.0	Pentium II 400 MHz
BATA (Tarantilis et al., 2002a)	6.5	Pentium 233 MHz
Best-SEPAS (Tarantilis, 2005)	6.6	Pentium II 400 MHz
LBTA (Tarantilis et al., 2002b)	6.8	Pentium 233 MHz
RSD (Reimann et al., 2002)	7.7	Pentium 900 MHz
AGES best (Cordeau et al., 2005; Mester and Braysy, 2007)	7.72	Pentium IV 2 GHz
UTSA (Cordeau et al., 2002)	13.8	Sun UltraSparc 10
BHS (Bullnheimer et al., 1999)	18.4	Pentium 100 MHz
Stand-HGA (Berger and Barkaoui, 2003)	21.3	Pentium 400 MHz
PTC (Rego, 2001)	24.65	4 Sun Sparc
Osman TS (Osman, 1993)	26.1	VAX 8600
BAGA (Baker and Ayechew, 2003)	29.1	Pentium 266 MHz
St-TABUROUTE (Gendreau et al., 1994)	46.8	Silicon Graphics 36 MHz
Barbarosoglu (Barbarosoglu and Ozgur, 1999)	56.2	Pentium 133 MHz
XuKelly (Xu and Kelly, 1996)	131.6	DEC ALPHA Workstation
Osman SA (Osman, 1993)	151.36	VAX 8600

Search, Xu Kelly, Granular Tabu Search, UTSA, Barbarosoglu, PTC, SEC). 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 (RT, BoneRoute, SEPAS). Simulated annealing (Osman Simulated Annealing) and threshold accepting algorithms (BATA, LBTA) are also applied efficiently in the VRP. Ant colony optimization for VRP have been proved to be very efficient and competitive with the other metaheuristics (RSD, D-Ants, BHS) just as genetic algorithms (Prins, BAGA, HGA). In Table 5, the ranking of all algorithms used for the comparisons and of the proposed algorithm is presented. The average quality of the solutions of all instances obtained by the HybPSO algorithm for the set of classic benchmark instances of Christofides is equal to 0.084%. The proposed algorithm is ranked among 39 algorithms in the *fifth* place. Table 6 presents only the ranking of the Nature Inspired methods used for the solution of the vehicle routing problem. From this Table it can be seen that the proposed nature inspired method (HybPSO) is ranked in the *first place* among all algorithms of this kind.

Although a fair comparison in terms of computational efficiency is difficult as the computational speed is affected, mainly, from the compiler and the hardware that are used, in Table 7 the average CPU time (in minutes) of the metaheuristic algorithms of the previous comparisons is presented. It can be observed from this Table that HybPSO is ranked in the second place among all metaheuristic algorithms.

5. Conclusions and future work

In this paper, a nature inspired approach was introduced for the effective handling of network analysis and optimization of logistics management problems. Specifically, a hybrid algorithmic nature inspired methodology was proposed, namely the HybPSO algorithm, for the effective handling of the vehicle routing problem.

One of the main contributions of this paper is to show that the particle swarm optimization can be used in hybrid synthesis with other metaheuristics for the solution of the vehicle routing problem with remarkable results both to quality and computational efficiency.A second contribution is the utilization of the MPNS–GRASP procedure for the generation of the initial particles. 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). HybPSO uses the path relinking strategy instead of the classic way that usually the particles move from their current solution to the local optimum or to the global optimum. Finally, the expanding neighborhood search strategy is utilized in order to improve the solutions of each particle in the swarm and to reduce the computational time of the algorithm. The algorithm was applied in a 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 0.084% and, thus, the algorithm is ranked in the fifth place among the thirty nine most known algorithms and in the first place among the nature inspired methods used for the solution of the VRP in the literature. In the future we will apply this algorithm in other variants of the classic vehicle routing problem, like the open vehicle routing problem, the vehicle routing problem with time windows and the stochastic vehicle routing problem.

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.
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.
Barbarosoglu, G., Ozgur, D., 1999. A tabu search algorithm for the vehicle routing problem. Computers and Operations Research 26, 255–270.
Berger, J., Barkaoui, M., 2003. A hybrid genetic algorithm for the capacitated vehicle routing problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, Chicago, pp. 646–656.
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: Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (Eds.), Combinatorial Optimization. Wiley, Chichester.

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: Langevine, A., Riopel, D. (Eds.), Logistics Systems: Design and Optimization. Wiley and Sons, New York, USA, pp. 279–298.

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. Springer, Heidelberg.

De Castro, L.D., Timmis, J., 2002. Artificial Immune Systems. A New Computational Intelligence Approach. Springer, Heidelber.

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. A Bradford Book. The MIT Press Cambridge, Massachusetts, London, England.

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

Fisher, M.L., 1995. Vehicle routing. In: Ball, M.O., Magnanti, T.L., Momma, C.L., Nemhauser, G.L. (Eds.), Network Routing, Handbooks in Operations Research and Management Science, vol. 8. North-Holland, Amsterdam, pp. 1–33.

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. John Wiley and Sons, New York.

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: Aarts, E.H.L., Lenstra, J.K. (Eds.), Local Search in Combinatorial Optimization. Wiley, Chichester, pp. 311–336.

Gendreau, M., Laporte, G., Potvin, J.Y., 2002. Metaheuristics for the Capacitated VRP. In: Toth, P., Vigo, D. (Eds.), The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, USA, pp. 129–154.

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: Glover, F., Kochenberger, G.A. (Eds.), Handbook of Metaheuristics. Kluwer Academic Publishers, Boston, pp. 1–36.

Golden, B.L., Assad, A.A., 1988. Vehicle Routing: Methods and Studies. North-Holland, Amsterdam.

Hansen, P., Mladenovic, N., 2001. Variable neighborhood search: principles and applications. European Journal of Operational Research 130, 449–467.

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

Kennedy, J., Eberhart, R., 1997. A discrete binary version of the particle swarm algorithm. Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics 5, 4104–4108.

Kennedy, J., Eberhart, R., Shi, Y., 2001. Swarm Intelligence. Morgan Kaufmann Publisher, San Francisco.

Laporte, G., Semet, F., 2002. Classical heuristics for the capacitated VRP. In: Toth, P., Vigo, D. (Eds.), The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, USA, pp. 109–128.

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.

Lichtblau, D., 2002. Discrete optimization using mathematica. In: Callaos, N., Ebisuzaki, T., Starr, B., Abe, J.M., Lichtblau, D. (Eds.), World Multi Conference on Systemics, Cybernetics and Informatics (SCI 2002), vol. 16, International Institute of Informatics and Systemics, pp. 169–174.

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

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

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 Lagrangian relaxation for the traveling salesman problem. Journal of Combinatorial Optimization 10, 311–326.

Marinakis, Y., Migdalas, A., Pardalos, P.M., 2007. 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., Migdalas, A., Pardalos, P.M., 2009. Multiple phase neighborhood search GRASP based on Lagrangian relaxation and random backtracking Lin Kernighan for the traveling salesman problem. Journal of Combinatorial Optimization 17, 134–156.

Mester, D., Braysy, O., 2007. Active guided evolution strategies for the large scale vehicle routing problems. Computers and Operations Research 34 (10), 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.

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

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

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

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.

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, New York, pp. 1317–1326.

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.

Rennard, J.F., 2006. Handbook of Research on Nature Inspired Computing for Economics and Management. Idea Group Reference, Hershey, PA, USA.

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

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., 2002. BoneRoute: an adaptive memory-based method for effective fleet management. Annals of Operations Research 115 (1), 227–241.

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.

Toth, P., Vigo, D., 2002. The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, USA.

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.