J Glob Optim (2007) 38:555–580
DOI 10.1007/s10898-006-9094-0

ORIGINAL ARTICLE

A new bilevel formulation for the vehicle routing problem and a solution method using a genetic algorithm

Yannis Marinakis · Athanasios Migdalas · Panos M. Pardalos

Received: 7 September 2006 / Accepted: 7 September 2006 / Published online: 27 October 2006

Abstract The Vehicle Routing Problem (VRP) is one of the most well studied problems in operations research, both in real life problems and for scientific research purposes. During the last 50 years a number of different formulations have been proposed, together with an even greater number of algorithms for the solution of the problem. In this paper, the VRP is formulated as a problem of two decision levels. In the first level, the decision maker assigns customers to the vehicles checking the feasibility of the constructed routes (vehicle capacity constraints) and without taking into account the sequence by which the vehicles will visit the customers. In the second level, the decision maker finds the optimal routes of these assignments. The decision maker of the first level, once the cost of each routing has been calculated in the second level, estimates which assignment is the better one to choose. Based on this formulation, a bilevel genetic algorithm is proposed. In the *first level* of the proposed algorithm, a genetic algorithm is used for calculating the population of the most promising assignments of customers to vehicles. In the *second level* of the proposed algorithm, a Traveling Salesman Problem (TSP) is solved, independently for each member of the population and for each assignment to vehicles. The algorithm was tested on two sets of benchmark instances and gave very satisfactory results. In both sets of instances the average quality is less than 1%. More specifically in the set with the 14 classic instances proposed by Christofides, the quality is 0.479% and in the second set with the 20 large scale vehicle routing problems, the quality is 0.826%. The algorithm is ranked in the tenth place among the 36 most known and effective

Y. Marinakis (✉) · A. Migdalas
Decision Support Systems Laboratory, Department of Production Engineering and Management, Technical University of Crete, 73100 Chania, Greece
e-mail: marinakis@ergasya.tuc.gr

A. Migdalas
e-mail: sakis@verenike.ergasya.tuc.gr

P. M. Pardalos
Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
e-mail: pardalos@cao.ise.ufl.edu

algorithms in the literature for the first set of instances and in the sixth place among the 16 algorithms for the second set of instances. The computational time of the algorithm is decreased significantly compared to other heuristic and metaheuristic algorithms due to the fact that the Expanding Neighborhood Search Strategy is used.

Keywords Vehicle routing problem · Metaheuristics · Bilevel programing · Genetic Algorithm

1 Introduction

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. Each customer must be assigned to exactly one of the vehicles and the total size of deliveries for customers assigned to each vehicle must not exceed the vehicle capacity. 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.

We can say that the problem arises as a generalization of the traveling salesman problem (TSP). The TSP requires the determination of a minimal cost cycle that passes through each node in a relevant graph exactly once. If costs are symmetric, that is, if the cost of traveling between two locations does not depend on the direction of travel, we have a symmetric TSP otherwise we have an asymmetric TSP. The multiple TSP arises if many salesmen or vehicles in the fleet are to leave from and return to the same depot. There are no restrictions on the number of nodes that each vehicle must visit except that each vehicle must visit at least one node. The most important variants of the VRP can be found in the following [5,16,26,44].

As it can be observed from the large number of variants of the VRP when one has to solve a real life VRP the first issue to deal with is the efficient formulation of the problem. The efficient formulation together with a powerful and clever algorithm (usually metaheuristic) based on the properties of the model gives to the decision maker (or to the researcher) the opportunity to deal with a number of real life problems and to find a near—optimum solution without spending excessive computational time. Here, the CVRP will be considered. In brief, the formulations of the CVRP are divided in three different categories, *Vehicle Flow Models*, *Commodity Flow Models*, and *Set Partitioning Models*. The *Vehicle Flow Models* use integer variables associated with each arc [5,7,10]. They can easily be used when the cost of the solution can be expressed as the sum of the costs associated with each arc. In the *Commodity Flow Models*, additional integer variables are associated with the arcs that express the flow of the commodities along the paths traveled by the vehicles [45]. Finally, in the *Set Partitioning Formulation* [45], a collection of circuits with minimum cost is determined which serves each customer once and, possibly, satisfies additional constraints. One of the main drawbacks of the last models is the huge number of variables. The explicit generation of all constraints is normally impractical, and one has to apply a column generation approach to solve the linear programing relaxation of the last models.

One of the most important formulations that has ever been proposed for the solution of the problem is the formulation of Fisher and Jaikumar [10]. This formulation

belongs to the category of vehicle flow models. Fisher and Jaikumar proved that the constraints of the problem can be separated in two sets. The first set of constraints are the constraints of a generalized assignment problem and they ensure that each route begins and ends at the depot, that every customer is served by some vehicle, and that the load assigned to a vehicle is within capacity. The second set of constraints corresponds to the constraints of a TSP for all customers of each vehicle. So, they solved a generalized assignment problem approximation of the VRP in order to obtain an assignment of customers. Subsequently, the customers assigned to each vehicle can be sequenced using any Traveling Salesman algorithm.

The formulation of Fisher and Jaikumar gave us the idea that the VRP can be formulated as a problem of two decision phases or as a problem of two decision levels. By saying two levels it is meant that in each level a different problem is solved, but the solution of the one level depends on the solution of the other. In particular it is assumed that the decisions in the second level are reacting to the decisions of the first level and that the decisions in the first level must be made by taking this fact into consideration. This kind of formulation is called bilevel formulation. Thus, the VRP can be viewed as a bilevel decision problem where in the first level decisions must be made concerning the assignment of the customers to the routes and in the second level decisions must be made concerning the routing of the customers.

A bilevel programing problem describes a hierarchical system which is composed of two levels of decision makers [30]. The higher level decision maker, known as *leader*, controls the decision variables y, while the lower level decision maker, known as *follower*, controls the decision variables x. The interaction between the two levels is modeled in their respective *loss functions* $\varphi(x, y)$ and $f(x, y)$ and often in the feasible regions. The leader and the follower play a *Stackelberg duopoly game*. The idea of the game is as follows: the first player, the leader, chooses y to minimize the loss function $\varphi(x, y)$, while the second player, the follower, reacts to leader's decision by selecting an admissible strategy x that minimizes his loss function $f(x, y)$. Thus, the follower's decision depends upon the leader's decision, i.e. $x = x(y)$, and the leader is in full knowledge of this. The general bilevel programing problem is stated as follows:

$$\text{(BP)} \quad \min_{y \in Y} \quad \varphi(x(y), y), \tag{1}$$

$$\text{subject to} \quad \psi(x(y), y) \leq 0, \tag{2}$$

$$\text{where} \quad x(y) = \arg\min_{x \in X} f(x, y), \tag{3}$$

$$\text{subject to} \quad g(x, y) \leq 0, \tag{4}$$

where $X \subset R^n$ and $Y \subset R^m$ are closed sets, ψ: $X \times Y \to R^p$ and g: $X \times Y \to R^q$ are multifunctions, φ and f are real-valued functions.

The *upper level* (1) and (2) corresponds to the leader, while the *lower level* (3) and (4) corresponds to the follower. The set $\mathcal{S} = \{(x, y) : x \in X, y \in Y, \psi(x, y) \leq 0, g(x, y) \leq 0\}$ is the *constraint set* of BP. For fixed $y \in Y$, the set $X(y) = \{x \in X : g(x, y) \leq 0\}$ is the *feasible set* of the follower. The set $\mathcal{R}(y) = \{x \in X : x \in \arg\min_{w \in X(y)} f(w, y)\}$ is called the *rational reaction set* of BP. The *feasible set* of BP is $\mathcal{F} = \{(x, y) \in \mathcal{S} : x \in \mathcal{R}(y)\}$. A feasible point $(x^\star, y^\star) \in \mathcal{F}$ is a *Stackelberg equilibrium* (with the first player as the leader) if $\varphi(x^\star, y^\star) \leq \varphi(x, y)$ for all $(x, y) \in \mathcal{F}$.

In the VRP, there is only one decision maker, although if we study the problem from a specific view the leader might be considered as a decision maker of the company (decision maker of the first level) which assigns clusters of customers and the

follower might be considered as the dispatcher who chooses the optimum route for servicing the customers in each cluster. This description is the key idea in the proposed formulation. Two decisions must be made although there is probably only one true decision maker. In the first phase (level), the decision maker assigns customers to the vehicles, checking the feasibility of the constructed routes (vehicle capacity constraints) without taking into account the sequence according to which the vehicle will visit the customers (routing). The main interest of the first level decision maker is to find all the promising assignments while the decision maker of the second level finds the optimal routes of these assignments. The decision maker of the first level estimates which is the best assignment based on the routing cost of the second level.

Based on such a two level formulation, a two level hybrid genetic algorithm is proposed. In the *first level* of the proposed algorithm a genetic approach similar to Hyb-GEN [28] is used in order to identify the most promising assignments of customers in vehicles. In particular the initial population is created by applying the MPNS-GRASP [29] and two local search algorithms are applied in order to improve the cost of the individuals. The selected algorithms for this phase are the shift and swap algorithms. In the *second level* of the proposed algorithm, a TSP is solved independently for each member of the population and for each assignment in vehicles. For the calculation of the routing, initially the MPNS-GRASP algorithm is used and subsequently the Expanding Neighborhood Search (ENS) method is applied. Once the decision maker of the first level is informed about the reaction of the follower he decides which of the solutions will consist the population of the next generation. This procedure is continued until convergence of the genetic algorithm or a prespecified maximum number of iterations is reached.

This paper is organized as follows. In Sect. 2 the most important algorithms for the solution of VRP are presented, in Sect. 3 the new proposed formulation is analyzed and a detailed description of the proposed genetic algorithm is given in Sect. 4. In Sect. 5, the computational results are presented and, finally, in the last section the conclusions and the future research are given.

2 Algorithms for the solution of the capacitated vehicle routing problem

The VRP was first introduced by Dantzig and Ramser in 1959 (see Bodin et al. [5]). 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 15 years. In the 1960s and 1970s the first attempts to solve the VRP 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 [4]. 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 [4]. In

this category also belongs the methods called route first cluster second in which first a giant TSP tour is constructed and then this route is decomposed into feasible vehicle routes. In the 1980s a number of mathematical programing procedures are proposed for the solution of the problem. One of these procedures is the algorithm proposed by Fisher and Jaikumar [10].

In the 1990s a number of algorithms, known as metaheuristics, that simulate physical phenomena, were applied for the solution of the VRP. 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. 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. 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. 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. 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. The reader can find more detailed descriptions of these algorithms in the survey papers [4,5,11,14,15,20,21,40].

3 Bilevel formulation for the vehicle routing problem

Let $G = (V, E)$ be a graph where V is the vertex set and E the arc set. The customers are indexed $i = 2, \ldots, n$, $j = 1, \ldots, n$ and $i = 1$ refers to the depot. The vehicles are indexed $k = 1, \ldots, K$. The capacity of vehicle k is Q_k. If the vehicles are homogeneous, the capacity for all vehicles is equal and denoted by Q. A demand q_j and a service time st_j are associated with each customer node j. The travel cost between customers i and j is c_{ij}. The problem is to construct a low cost, feasible set of routes—one for each vehicle (starting and finishing at the depot). A route is a sequence of locations that a vehicle must visit along with the indication of the serve it provides [5]. The vehicle must start and finish its tour at the depot. Customer orders cannot be split. The customers are assigned to a single route until the routes reach capacity or time limits, subsequently a new customer is selected as a seed customer for the new route and the process is continued. A seed customer is a customer not yet assigned in a route that is used in order to initialize a new route. The distance of the k seed customer from the depot is d_k.

In order to present the bilevel model for the problem, we define the following variables:

$$z_k = \begin{cases} 1, & \text{if } k \text{ is a seed customer,} \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{kj} = \begin{cases} 1, & \text{if customer } j \text{ belongs in the same route,} \\ & \text{with the seed customer } k, \\ 0, & \text{otherwise} \end{cases}$$

and

$$y_{ij} = \begin{cases} 1, & \text{if edge } (i,j) \text{ is in the route,} \\ 0, & \text{otherwise.} \end{cases}$$

The bilevel formulation for the VRP is then:

$$\text{(leader)} \quad \min_{x,z} \sum_{k=1}^{K} d_k z_k + \sum_{k=1}^{K} \sum_{j=1}^{n} c_{kj} x_{kj} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} y_{ij}, \tag{5}$$

s.t.

$$\sum_{k=1}^{n} z_k = K, \tag{6}$$

$$\sum_{j=1}^{n} q_j x_{jk} \le (Q - q_k) z_k, \quad \forall k \in K, \tag{7}$$

$$\sum_{k=1}^{K} x_{kj} = 1, \quad \forall j = 1, \ldots, n, \tag{8}$$

where

$$\text{(follower)} \quad \min_{y|x,z} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} y_{ij}, \tag{9}$$

s.t.

$$y_{ij} \le x_{ik}, \quad i, j = 1, \ldots, n, \quad k = 1, \ldots, K, \tag{10}$$

$$\sum_{i=1}^{n} y_{ij} = 1, \quad j = 1, \ldots, n, \tag{11}$$

$$\sum_{j=1}^{n} y_{ij} = 1, \quad i = 1, \ldots, n, \tag{12}$$

$$\sum_{j \in V} \sum_{i \in V} y_{ij} \le |S| - 1, \quad \forall S \subset V, S \neq \emptyset. \tag{13}$$

The objective function of the leader (5) minimizes the sum of the seed customers' costs from depot, the sum of the costs of assigning customers to the routes, i.e. the assignment of the customers to the seed customers and the routing cost. Constraints (6) requires z_k to be set equal to the number of vehicles. Constraints (7) are the vehicle capacity constraints. Finally, constraints (8) state that every customer j must be on exactly one route (vehicle). The objective function of the follower (9) describes the routing cost of each vehicle based on the assignment by the leader. Constraints (10) require that the TSP should be solved only for the group of customers that have been assigned to the specific vehicle. Constraints (11) and (12) are degree constraints specifying that each node is entering exactly once and is leaving it exactly once. Constraints (13) are subtour elimination constraints.

4 Heuristic algorithm for the solution of the Bilevel vehicle routing problem

4.1 General description of hybrid genetic algorithm

The outline of the two level hybrid algorithm VRPBilevel (VRPB) for the solution of the problem as stated in the previous section is presented in the following:

Initialization

First level problem

1. Create the initial population of *NR* individuals.
2. Evaluate the fitness of each individual.
3. Improve the fitness of each individual via a local search strategy.

Second level problem

1. For each individual of the initial population solve a TSP.
2. Initial computation of lower bounds.
3. Improve the solution of each individual using the ENS Method.

Main algorithm

1. Set the number of generations equal to zero.
2. Do while stopping criteria are not satisfied (the maximum number of generations has not been reached or the convergence of the genetic algorithm has not occurred):
   First level problem
   - 2.1.1 Select the parents from the current population and choose via roulette wheel selection the pairs for mating.
   - 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 a pure greedy procedure.
   - 2.1.3 Improve each offspring by mutation operator and insert the resulting offspring to the new population.
   - 2.1.4 Repeat the previous three steps until all parents are selected and mated.

   Second level problem
   - 2.2.1 For each individual of the initial population solve a TSP using MPNS-GRASP algorithm.
   - 2.2.2 Update lower bounds.
   - 2.2.3 Improve the solution of each individual using ENS Strategy.

   Population replacement
   - 2.3 Rank the offsprings and the parents via their fitness function and the reaction of the follower and select for the new population a number of individuals equal to the initial population and proceed to the next generation.
3. enddo
4. Return the best individual.

4.2 The First Level Problem (Leader Problem)

4.2.1 Initial population

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) is used to initialize the population.

The GRASP [27,37] 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 generalized assignment 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 (c.f. multi-start local search). The pseudocode of the algorithm is presented next:

algorithm GRASP
do while stopping criteria not satisfied
    call GREEDY_RANDOM_SOLUTION(Solution)
    call LOCAL_SEARCH(Solution)
    if Solution is better than Best_Solution_Found **then**
        Best_Solution_Found ⟵ Solution
    endif
enddo
return Best_Solution_Found

4.2.2 Greedy algorithm for the generalized assignment problem

In this method, an assignment of the customers to vehicles based on the solution of the Generalized Assignment Problem (GAP) is realized and, then, a TSP is solved for each assignment in order to find the sequence in which the vehicle should visit the customers. Initially, the minimum number of vehicles is found that minimizes the optimal GAP value, using the function "Selection of Seed Customers". This value is set equal to the minimum number of seed customers:

Selection of seed customers

$TotDemand = \sum_{j=1}^{n} q_j$! q_j = demand of each customer

!TotDemand = the total demand of all customers

$k_{min} = \frac{TotDemand}{Q}$! Q = capacity of the vehicles,

```
!k_min = minimum number of vehicles
k = 0
do j = 1, n
    unused(j) = 0  !  unused shows if a customer is used in a route
enddo
do j = 1, n
    if d_j > Q/2 then
        k = k + 1, seed(k) = j, route(k, 1) = i, unused(j) = 1
    endif
enddo
if k < k_min then
    call Make_Queue(V)
    κ1 = 0
    do while κ1 < D  !D is the size of the RCL
        call Select_Max_From_List((i, j), E)
        !i = a customer and j = a seed customer
        call Delete_Max_From_List((i, j), E)
        Add i to RCL
        κ1 = κ1 + 1
    enddo
    do while k < k_min
        Select i randomly from RCL
        k = k + 1
        seed(k) = i
        route(k, 1) = i
        unused(j) = 1
    enddo
endif
return Initial Seed Customers
```

The function *Make_Queue* converts an array into a heap. The function which is used to select the root of an increasing order heap is *Select_Max_From_List*, while the function *Delete_Max_From_List* is used to remove the root of the heap and adjust the list. The Restricted Candidate List (RCL) (see Sect. 4.3.1) is created taking into account the distances of the non-assigned customers from the seed customers and the depot. If the number of seed customers is less than the number of vehicles, the rest of the seed customers are selected from the RCL. Then, one non-assigned customer is selected randomly from the RCL. The assignment cost is calculated of the customer in each route, checking if the two constraints are not violated, and, then, the customer is assigned to the best feasible route using the function "Assignment of Customers":

```
Assignment of customers
do  m = 1, k_min
    route_cap(m) = q_seed(m)  ! route_cap = the capacity of each route
    route_length(m) = servicetime + cost(m, 0)
    !route_length = the length of each route customer 0 is the depot
call Make_Queue(V)
κ2 = 0
do while κ2 < D   ! D = is the size of the RCL
```

call `Select_Min_From_List`((i,j),E)
! i = a customer, and j = a seed customer
call `Delete_Min_From_List`((i,j),E)
Add i to RCL1
$\kappa_2 = \kappa_2 + 1$
enddo
do while $unused(j) = 1, \forall j \in V$
Select i randomly from RCL
Delete i from RCL
Order cost(i,seed)
! Order customers based on the distance from the seed customers
$test = 0$
do $m = 1, k_{min}$
if $q_i + route\_cap(m) \leq Q$ and
$servicetime + route\_length(m) \leq MaxRoute$ **then**
!$MaxRoute$ = the maximum route length restriction
Add i in $route(m, i)$
$route\_cap(m) = q_i + route\_cap(m)$
$route\_length(m) = servicetime + route\_length(m)$
$unused(i) = 1$
$test = 1$
endif
enddo
if $test = 0$ **then**
$k_{min} = k_{min} + 1$
$seed(k_{min}) = j$
route(k_{min},1) = i
unused(i) = 1
$route\_cap(k_{min}) = q_{seed(k_{min})}$
$route\_length(k_{min}) = servicetime + cost(k_{min}, 0)$
endif
enddo
return Assignment in vehicles

The function which is used to select the root of an increasing order heap is *Select_Min_From_List*, while the function *Delete_Min_From_List* is used to remove the root of the heap and adjust the list.

4.2.3 Calculation of fitness value and selection probability

The fitness of each individual is related to the assignment in each route. Since the generalized assignment problem is a minimization problem, the objective function value is inversely proportional to the fitness value of the solution. Therefore, high-relative objective value corresponds with low-fitness value, and low-relative objective corresponds with high-fitness value. A way to accomplish this is to find initially the assignment in the population with the maximum cost and to subtract from this value the cost of each of the other assignments. By doing this, the higher fitness value corresponds to an assignment with the shorter cost. Since the probability of selecting an

individual for mating is related to its fitness and since the individual with the worst assignment has fitness equal to zero, it will never be selected for mating. Therefore in order to avoid its total exclusion the fitness of all individuals in this population is incremented by one resulting thus in the worst individual having a fitness of one. The parents are selected for mating via proportional selection, also known as roulette wheel selection [28].

4.2.4 Crossover

We propose a complex crossover operator, the dog breed crossover, which initially identifies the common characteristics of the parent individuals and, then, inherits them to the offsprings. Subsequently, a greedy procedure is applied to each offspring in order to complete the assignment.

The common characteristics are used in the offsprings because if two or more solutions of a combinatorial problem have common characteristics there is a high-probability that these, also, belong to the optimal solution of the problem. Moreover, the inheritance of good characteristics from highly adapted parents may produce even fitter offspring.

4.2.5 Mutation

Mutation operators for the GAP are synonymous to local exchange heuristics. The most known of these heuristics are the shift algorithm and swap algorithm, where both of them are used in this algorithm. In the first (shift algorithm), S' is obtained by changing the assignment of one customer from the initial solution S. In the second (swap algorithm), S' is obtained by exchanging the assignment of two customers from the initial solution S. These are described by the following pseudocode:

```
Mutation phase for GAP
! Shift Algorithm
do while no further improvement
    Select i for route r1
    Select the candidate for inclusion route (different from j)
    if Capacity and Max Route constraints not violated then
        if Cost_new_assignment < Cost_old_assignment then
            Change the route of i
            Cost_old_assignment = Cost_new_assignment
        endif
    endif
enddo
! Swap Algorithm
do while no further improvement
    Select i for route r1
    Select j for route r2
    if Capacity and Max Route constraints not violated then
        if Cost_new_assignment < Cost_old_assignment then
            Exchange customers i and j
            Cost_old_assignment = Cost_new_assignment
        endif
```

endif
enddo
return Assignment in vehicles

4.3 The second level problem (follower problem)

4.3.1 Pure greedy algorithm for the TSP

In the follower problem, the modified GRASP algorithm is also applied for the solution of the TSP. Initially, a 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, as it was presented above. From this list, the first D edges are selected in order to form the RCL. This type of RCL is called cardinality based RCL. 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. Once an element has been chosen for inclusion in the tour, a tour construction heuristic is applied in order to insert it in the partial tour.

The tour construction heuristic is a pure greedy algorithm. Initially, the degree of all the nodes is set equal to zero. Then, one edge at a time is selected randomly from the RCL. Each edge (i, j) is inserted in the partial route taking into account that the degree of i and j should be less than or equal to 2 and the new edge should not complete a route with fewer than n vertices. In this algorithm, a number of paths are created which are finally joined to a single route [25] . A candidate edge is added to the solution if and only if it does not create a subtour that excludes a node that should belong to the tour.

In the following, a pseudocode of the pure greedy algorithm is presented:

Pure greedy algorithm
$S = \emptyset$! S is the current solution
Initial values for $degree(i_1) = 0 \; \forall \; i_1$
call `Make_Queue`(E)
call `Init_Set`(parent, number of nodes)
$\kappa = 0$
do while $\kappa < D$! D = the size of the RCL
call `Select_Min_From_List`((i,j),E)
call `Delete_Min_From_List`((i,j),E)
Add $l = (i, j)$ to RCL
$\kappa = \kappa + 1$
enddo
$\kappa_1 = 0$
do while $degree(i_1) \neq 2, \; \forall i_1 = 1, \ldots n$
Select $l = (i, j)$ randomly from RCL
Delete $l = (i, j)$ from RCL
if $degree(i) \neq 2$ and $degree(j) \neq 2$ **then**
if $\kappa_1 < n - 1$ **then**
root1 = `Collapsing_Find_Set`(i,parent)

```
                root2 = Collapsing_Find_Set(j,parent)
                if root1 ≠ root2 then
                        call Merge_Paths((i, j), S)
                        call Union(root1,root2,parent)
                        degree(i) = degree(i) + 1
                        degree(j) = degree(j) + 1
                        κ1 = κ1 + 1
                endif
        else
                call Merge_Paths((i, j), S)
                degree(i) = degree(i) + 1
                degree(j) = degree(j) + 1
                κ1 = κ1 + 1
        endif
    endif
    Add the (D + m)th edge from list to the RCL
enddo
return S
```

The function *Init_Set* initializes disjoint sets, where each set has one element (node) and the total number of sets is equal to the number of nodes. The function *Collapsing_Find_Set* returns the root of the set to which the element belongs. At the same time, this function collapses the tree in order to reduce the longest path in the tree, giving, thus, a better complexity for a sequence of searches. The function *Union* joins two disjoint sets with roots 1 and 2, where roots 1 and 2 are values returned from the function *Collapsing_Find_Set*, by making the smallest set a subtree of the other set. Finally, the function *Merge_Paths* merges the two different paths if roots 1 and 2 are not the same.

4.3.2 Calculation of lower bounds

For a given set of nodes, an 1-tree ([19]) is a tree connecting the node set $\{2, 3, \ldots, n\}$, and having in addition two distinct arcs connecting to node 1. Therefore, an 1-tree is a connected graph with one cycle. The weight of the 1-tree is the sum of the cost of all its arcs. A minimum weight tree can be constructed by finding a minimum spanning tree for the node set $\{2, 3, \ldots, n\}$, and by adding to it the two arcs of minimum cost incident to node 1. Any TSP can be described as an 1-tree tour in which each node has degree 2. Thus, the minimum weight 1-tree implies a lower bound (LBD) on the length of the optimal traveling salesman tour.

The TSP can be formulated as follows:

$$\text{(TSP) min cost(T)} \quad \text{where} \quad \begin{cases} T \text{ is an 1-tree with root node 1} \\ degree(i) = 2 \ \ \forall \text{ node } i \text{ except the root.} \end{cases} \tag{14}$$

The 1-tree Lagrangian Relaxation Subproblem (SUB) is obtained by relaxing the degree constraints, that is,

$$\mathrm{SUB}(\lambda_i) = \min \mathrm{cost(T)} - \sum_{i \in N-\{1\}} (\mathrm{degree}(i) - 2)\lambda_i, \tag{15}$$

where T is an 1-tree with root node 1 and the $\lambda_i, i \neq 1$, are the Lagrangian multipliers. The optimal values of the λ_i for all nodes $i \neq 1$ are calculated using the subgradient algorithm:

$$\lambda_i^{(\kappa+1)} = \lambda_i^{(\kappa)} + \alpha^{(\kappa)} \frac{\mathrm{UBD}^{(\kappa)} - \mathrm{LBD}^{(\kappa)}}{||\mathbf{g}||^2} (\mathrm{degree}(i) - 2), \tag{16}$$

where $\mathbf{g} = [g(i)]_{i \in V-\{1\}}$, with $g(i) = \mathrm{degree}(i) - 2$ denotes the subgradient, κ is the iteration counter, UBD and LBD are upper and lower bounds in the optimal TSP objective value, and α is a parameter in (0,2). The following pseudocode describes the approach.

Calculation of lower bounds
$\kappa = 0$
Initial values for $\lambda_i^{(\kappa)} = 0$
do while (Max number of iterations reached or $e \leq$ threshold value)
    Solve $SUB(\lambda_i^{(\kappa)})$
    The cost of the SUB is a new lower bound for the TSP (NLBD)
    if $(degree(i) = 2)\ \forall$ *node* **then**
        The cost of the SUB is an upper bound
        The algorithm stops with the optimal solution to
        the Lagrangian dual problem
    else
        Adjust the solution with Christofides' algorithm [22]
        The cost produced by Christofides' algorithm is
        a new upper bound (NUBD)
        Improve NUBD using the 2-opt algorithm
    endif
    if NUBD $<$ UBD **then**
        UBD = NUBD
    endif
    if NLBD $>$ LBD **then**
        LBD = NLBD
    endif
    $\kappa = \kappa + 1$
    $\lambda_i^{(\kappa+1)} = \lambda_i^{(\kappa)} + \alpha^{(\kappa)} \frac{UBD^{(\kappa)} - LBD^{(\kappa)}}{||\mathbf{g}||^2} (degree(i) - 2)$
    $e = \frac{UBD - LBD}{UBD} 100\%$
enddo

In each iteration of the algorithm, a minimum weight 1-tree is calculated and if the solution is feasible, i.e. the degree of all nodes including the root node is equal to 2, the algorithm stops. If a minimum weight 1-tree is a tour, with respect to non-modified

costs (i.e. $\lambda_i = 0$), then it is an optimal traveling salesman tour. When the solution is not feasible the cost of the 1-tree constitutes a lower bound and an effort is made to convert it into a feasible one with the use of the Christofides' heuristic [22]. In the resulting tour, a 2-opt heuristic is applied in order to improve it. The cost of this tour yields an upper bound. As it is desirable to have the lower and the upper bounds as good as possible, the subroutines of the main phase of the algorithm are called every κ_1, for instance $\kappa_1 = 10$, iterations. At the end of each iteration, the new cost of the 1-tree is compared to the current lower bound and if the cost is higher, then the lower bound is updated. Similarly, the upper bound is updated only if the current value is lower than the current upper bound. Thus, a sequence of lower and another of upper bounds are created which tend toward each other.

4.3.3 Expanding neighborhood search strategy for the TSP

The ENS has been proven very efficient for the solution of the TSP [27]. ENS is based on a method called Cycle Restricted Local Search Moves (CRLSM) and, in addition, it has a number of local search phases. In this implementation, the local search strategies used are 2-opt and 3-opt.

In the 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, i.e. the search is restricted to edges with one of their end-nodes inside a circle with radius length at most equal to the sum of the costs (lengths) of the two candidates for deletion edges.

In the following, a description of the CRLSM Strategy for a 2-opt trial move is presented. In this case, there are three possibilities based on the costs of the candidates for deletion and inclusion edges:

(1) If both new edges increase in cost, a 2-opt trial move can not 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).
(2) 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, can not 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).
(3) 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).

The ability of the algorithm to change between different local search strategies is another innovation of the proposed algorithm and overcomes the third obstacle. 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 [12] and recently by Hansen and Mladenovic [18]. 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

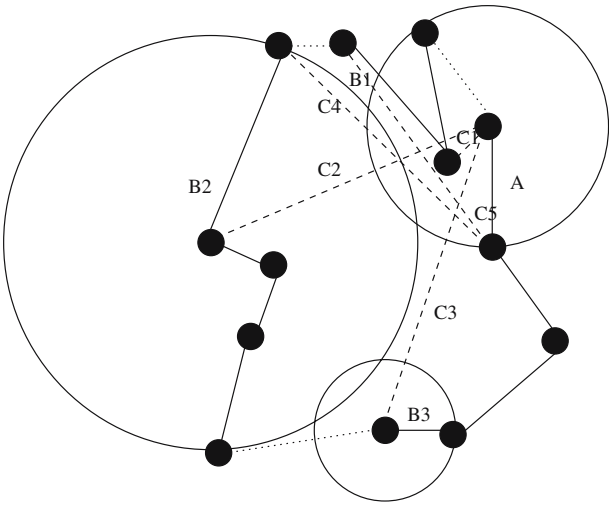


Fig. 1 Cycle restricted local search moves method

optimum. On the other hand, Hansen and Mladenovic proposed a more systematical method to change between different neighborhoods, called Variable Neighborhood Search.

The ENS 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 can not 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 main differences of ENS from the other two methods is the use of the CRLSM which restricts the search in circles around the candidates for deletion edges, the stopping criterion that is based on lower bounds to the optimal objective value, and, finally, the more sophisticated way that the local search strategy can be changed inside the circles.

The idea of searching inside a radius of the circle of the neighborhood search, the CRLSM Strategy, is the most innovative feature of the proposed algorithm. In ENS 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 CRLSM strategy, for example $s = A/2$, where A is the cost 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 cannot 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 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 ENS Method is presented. The following pseudocode describes this approach.

```
Expanding neighborhood search
s = A/2
m = 0  !m =index for the local search methods
do while (ω > e or s < A + B)   !ω = the quality of the solution
      m = 1
      Call first local search strategy
      if ω < e then
            STOP with the current solution
      else
            do while (m < M)
            ! M = the number of local search strategy
                  if ω < b1 or a local optimum is found then
                        Change local search method
                        m = m + 1
                        if ω < e then
                              STOP with the current solution
                        endif
                  endif
            enddo
      endif
      s = 1.1 * s
      Update lower bounds, b1 and e
enddo
```

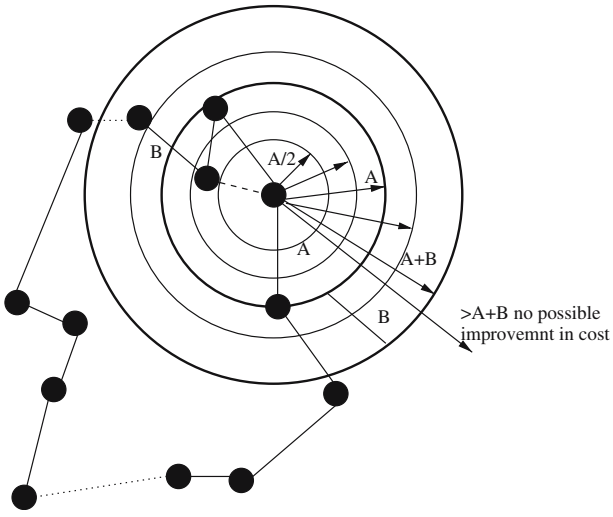
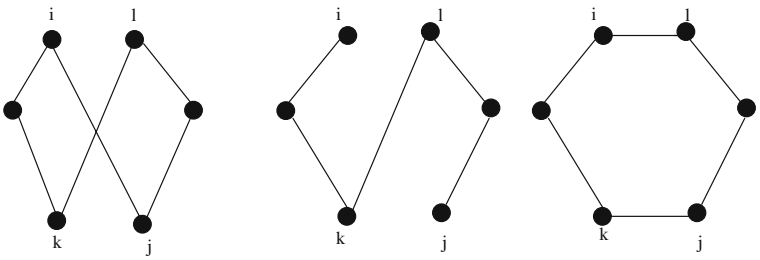


Fig. 2 Expanding neighborhood search

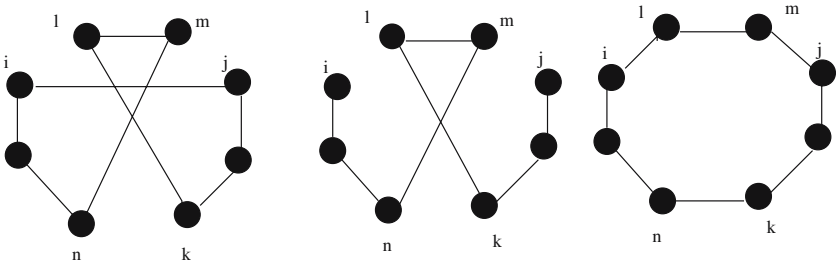
The two local search algorithms that are used in ENS strategy are 2-opt and 3-opt. First, the neighborhood function is defined as exchanging two edges of the current solution with two other edges. This procedure is known as 2-opt procedure and was introduced by Lin for the TSP [24]. Note that there is only one way to reconnect the paths. A restricted version of the approach is applied (Fig. 3). The following pseudocode describes this approach:

2-opt Strategy
Given S !S is the current solution
call `Make_Queue`(E_1) !E_1 = the set of arcs of the initial solution
do while No further improvement in the solution can be achieved
  call `Select_Max_From_List`$((i,j),E_1)$
  do for all possible (i_1,j_1) ! (i_1,j_1) = an edge of the current tour
    $S' = S \setminus (i,j) \setminus (i_1,j_1) \cup (i,i_1) \cup (j,j_1)$
    call `Calc_Cost`$(S', cost_{S'})$
    if $(cost_{S'} < cost_S)$ **then**
      Update the solution $S \longleftarrow S'$
      $cost_S = cost_{S'}$
      call Re_Arrange_List(S)
    endif
  enddo
enddo
return S

The function *Select_Max_From_List* is used for selecting the root of the decreasing ordering heap. The function *Re_Arrange_List*(S) calls the functions *Delete_From_List*



A) Example of 2-opt method



B) Example of 3-opt method

Fig. 3 Example of 2-opt and 3-opt

and *Add_to_List*, where the function *Delete_From_List* is used for removing the deleted edges of the heap and adjusting the list and the function *Add_to_List* is used for adding the selected edges in the heap and adjusting, again, the heap. The function *Calc_Cost* calculates the cost of the candidate tour.

For example, the function *Re_Arrange_List*(S) for the 2-opt strategy works as follows:

Re_Arrange_List(S)
call Delete_from_List(i, j)
call Delete_from_List(i_1, j_1)
call Add_to_List(i, i_1)
call Add_to_List(j, j_1)

The 3-opt heuristic (Fig. 3) is quite similar to the 2-opt. However, because it uses a larger neighborhood, it introduces more flexibility in modifying the current tour. The procedure is quite similar with 2-opt except that in the 3-opt strategy two edges are candidate for deletion from the tour.

4.4 Population replacement and termination process

During the course of evolution, natural populations evolve according to the principles of natural selection and survival of the fittest. Individuals who are more succesfull in adapting to their environment will have a better chance of surviving and reproducing, while individuals which are less fit have a higher probability of being eliminated. Initially, all the individuals of the population are sorted w.r.t. their fitness values and the solution gave the reaction of the follower (k is equal with the initial population). Subsequently, the k individuals, for example $k = 100$, with the best reaction of the follower will replace the old population. In the proposed algorithm, the algorithm stops using either the maximum number of generations or if there is genetic convergence, i.e. the individuals of the current population are identical.

5 Computational results for the vehicle routing problem

The VRPB algorithm was implemented in Fortran 90 and were compiled using the Lahey f95 compiler on a Pentium III at 667 MHz, running Suse Linux 8.2. The algorithms was tested on two sets of benchmark problems. The 14 benchmark problems proposed by Christofides and the 20 large scale vehicle routing problems proposed by Golden. 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 also have maximum route length restrictions but with zero service times.

The efficiency of the Bilevel Genetic—GRASP—ENS (VRPB) algorithm is also 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 $\omega = \frac{100(c_{\text{VRPBilevel}} - c_{\text{opt}})}{c_{\text{opt}}}\%$, where $c_{\text{VRPBilevel}}$ denotes the cost of the solution found by VRPBilevel and c_{opt} is the cost of the best known solution. It can be seen from Tables 1 and 2, that the

VRPBilevel 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 solution is between 0.64 and 1.31% with average quality 0.479%. For the 20 large scale vehicle routing problems, proposed by Golden, in which the algorithm was, also, tested, the quality of the solution is between 0.33 and 1.63%, with average quality 0.826%. These results denote the efficiency of the proposed algorithm.

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 VRP. 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 algorithms are the Clarke and Wright algorithm, the 1-petal and the 2-petal algorithms, the Fisher and Jaikumar algorithm, the Wark and Holt algorithm, the B-SL Gavish algorithm, the Christofides–Mignozzi–Toth algorithm, the Desrochers algorithm, the Sweep algorithm and, finally, the Mole and Jameson 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 (Tailard, TABUROUTE, Osman Tabu 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 meta-

Table 1 Results of VRPBilevel in Christofides benchmark instances

	Nodes	Capacity	Max. tour length	Service time	VRP-Bilevel	Optimum	Quality (%)	CPU (min)
1	51	160	∞	0	524.61	524.61	0.00	0.02
2	76	140	∞	0	835.26	835.26	0.00	0.23
3	101	200	∞	0	826.14	826.14	0.00	0.29
4	151	200	∞	0	1028.42	1028.42	0.00	0.83
5	200	200	∞	0	1306.17	1291.45	1.14	2.30
6	51	160	200	10	555.43	555.43	0.00	0.02
7	76	140	160	10	909.68	909.68	0.00	0.27
8	101	200	230	10	865.94	865.94	0.00	0.84
9	151	200	200	10	1177.76	1162.55	1.31	1.48
10	200	200	200	10	1404.75	1395.85	0.64	3.03
11	121	200	∞	0	1051.73	1042.11	0.92	0.20
12	101	200	∞	0	825.57	819.56	0.73	0.34
13	121	200	720	50	1555.39	1541.14	0.92	0.42
14	101	200	1040	90	875.35	866.37	1.03	0.31

Table 2 Results of VRPBilevel in large scale vehicle routing Instances

	Nodes	Capacity	Max. tour length	Service time	VRP-Bilevel	Optimum	Quality (%)	CPU (min)
1	240	550	650	0	5702.48	5644.02	1.03	1.48
2	320	700	900	0	8476.64	8447.92	0.34	1.77
3	400	900	1,200	0	11117.38	11036.22	0.74	5.08
4	480	1000	1,600	0	13706.78	13624.53	0.60	6.18
5	200	900	1,800	0	6482.67	6460.98	0.33	1.02
6	280	900	1,500	0	8501.15	8412.88	1.05	1.19
7	360	900	1,300	0	10254.35	10195.59	0.57	2.03
8	440	900	1,200	0	11957.15	11828.78	1.08	5.28
9	255	1,000	∞	0	589.12	585.43	0.63	1.07
10	323	1,000	∞	0	749.15	743.17	0.35	2.30
11	399	1,000	∞	0	934.24	923.17	1.20	2.75
12	483	1,000	∞	0	1138.92	1128.03	0.97	6.8
13	252	1,000	∞	0	868.80	865.01	0.43	2.74
14	320	1,000	∞	0	1096.18	1083.65	0.93	1.95
15	396	1,000	∞	0	1367.25	1351.35	0.98	6.22
16	480	1,000	∞	0	1645.24	1634.74	0.64	9.59
17	240	200	∞	0	711.07	708.74	0.33	2.15
18	300	200	∞	0	1015.12	998.83	1.63	2.16
19	360	200	∞	0	1389.15	1367.2	1.61	2.62
20	420	200	∞	0	1842.17	1822.94	1.06	4.36

heuristics (RSD, D-Ants, BHS) just as genetic algorithms (Prins, BAGA, HGA). In Tables 3 and 4, the ranking of all algorithms used for the comparisons and of the proposed algorithms is presented.

The average quality of the solutions of all instances obtained by the VRPBilevel algorithm for the set of classic benchmark instances of Christofides is equal to 0.479% for the VRPBilevel. The proposed algorithm is ranked among 36 algorithms in the tenth place. The VRPBilevel which is an algorithm based on a different modeling of the VRP using discrete bilevel optimization, is very fast and efficient and has all the good characteristics of the Hyb-GEN [28] as they are both genetic algorithms which use the same methods (ENS, MPNS-GRASP) in different phases of the algorithms. Thus, the VRPBilevel can be viewed as the beginning of a new approach for the solution of this kind of problems. For the large scale vehicle routing problems, the average quality of the solutions is equal to 0.826% for the VRPBilevel. The proposed algorithm is ranked among 16 algorithms in the sixth place the VRPBilevel. These results confirm our conclusions of the previous analysis.

It should be noted that a fair comparison in terms of computational efficiency is difficult because the computational speed is affected, mainly, from the compiler and the hardware that are used. Despite this fact, the average CPU time (in minutes) of the metaheuristic algorithms of the previous comparisons is presented in Tables 5 and 6 as such a comparison is necessary in order to show the computational efficiency of our algorithm. It can be seen from these Tables that VRPBilevel is the faster of all metaheuristic algorithms in the first set of instances and it is ranked in the second place among all algorithms in the large scale vehicle routing instances.

Table 3 Comparison of various heuristics and metaheuristics algorithms with VRP Bilevel in Christofides Instances

Rank	Algorithm	Quality
1	RT [38]	0.00
2	Tailard [39]	0.051
3	best-Prins [32]	0.085
4	Best-SEPAS [40]	0.182
5	St-SEPAS [40]	0.195
6	best-TABUROUTE [13]	0.198
7	BoneRoute [43]	0.226
8	stand-Prins [32]	0.235
9	RSD [33]	0.383
10	VRP-Bilevel	0.479
11	D-Ants [34]	0.481
12	stand-HGA [3]	0.485
13	LBTA [42]	0.498
14	BAGA [1]	0.504
15	BATA [41]	0.525
16	PTC [36]	0.549
17	Barbarosoglu [2]	0.57
18	Wark Holt [21]	0.635
19	Granular Tabu Search (GTS) [46]	0.64
20	UTSA [8]	0.689
21	st-TABUROUTE [13]	0.863
22	Osman Tabu Search [31]	1.011
23	BHS [6]	1.511
24	SEC [35]	1.539
25	Xu and Kelly [47]	1.718
26	Osman Simulated Annealing [31]	2.105
27	2-Petal [21]	2.430
28	Fisher and Jaikumar [10]	2.659
29	B-SL [21]	3.295
30	Gavish [45]	3.415
31	Christofides-Mignozzi-Toth [7]	5.055
32	1-Petal [21]	5.937
33	CW [21]	6.724
34	Desrochers [45]	6.922
35	Sweep [21]	7.177
36	Mole–Jameson [7]	10.906

6 Conclusions

In this paper, a new formulation for the VRP is proposed based on bilevel programing. This formulation separates the problem in two different problems, the generalized assignment problem for the assignment of the customers in vehicles, and the TSP for the routing of the vehicles. Based on this formulation, a bilevel genetic algorithm is proposed. In the first level of the proposed algorithm, a genetic algorithm is used for calculating the population of the most promising assignments of customers to vehicles. In the second level of the proposed algorithm, a TSP is solved, independently for each member of the population and for each assignment to vehicles. The algorithm was applied in two set of benchmark instances and gave very satisfactory results in both sets (average quality less than 1%). More specifically in the set with the classic benchmark instances, proposed by Christofides, the average quality is 0.479% and in the second set of benchmark instances, the 20 large scale vehicle routing problem, is 0.826% and, so, the algorithm is ranked in the tenth place among 36 most known

Table 4 Comparison of various heuristics and metaheuristics algorithms with VRP Bilevel in Golden Instances

Rank	Algorithm	Quality
1	best-D-Ants [34]	0.201
2	stand-SEPAS [40]	0.203
3	best-Prins [32]	0.515
4	BoneRoute [43]	0.529
5	Variable Record to Record Travel (VRTR) [23]	0.66
6	VRP-Bilevel	0.826
7	aver-D-Ants [34]	0.837
8	st-Prins [32]	0.927
9	LBTA [42]	1.545
10	BATA [41]	1.576
11	GTS [46]	2.473
12	LS [9]	3.123
13	Record To Record Travel (RTR) [17]	3.511
14	LT [9]	4.709
15	Xu-Kelly [47]	8.119
16	Golden [17]	12.487

Table 5 Running times of various heuristics and metaheuristics algorithms in Christofides Instances

Algorithm	CPU (min)	Computer used
VRP-Bilevel	0.76	Pentium III 667 MHz
SEC [35]	2.32	HP 9000/712
D-Ants [34]	3.28	Pentium 900 MHz
GTS [46]	3.84	Pentium 200 MHz
best-Prins [32]	5.2	Pentium 1,000 MHz
stand-Prins [32]	5.2	Pentium 1,000 MHz
St-SEPAS [40]	5.6	Pentium II 400 MHz
BoneRoute [43]	6.0	Pentium II 400 MHz
BATA [41]	6.5	Pentium 233 MHz
Best-SEPAS [40]	6.6	Pentium II 400 MHz
LBTA [42]	6.8	Pentium 233 MHz
RSD [33]	7.7	Pentium 900 MHz
UTSA [8]	13.8	Sun UltraSparc 10
BHS [6]	18.4	Pentium 100 MHz
stand-HGA [3]	21.3	Pentium 400 MHz
PTC [36]	24.65	4 Sun Sparc
Osman TS [31]	26.1	VAX 8600
BAGA [1]	29.1	Pentium 266 MHz
st-TABUROUTE [13]	46.8	Silicon Graphics 36 MHz
Barbarosoglu [2]	56.2	Pentium 133 MHz
Xu and Kelly [47]	131.6	DEC ALPHA Workstation
Osman SA [31]	151.36	VAX 8600

algorithms in the literature in the first set and in the sixth place among sixteen algorithms in the second set. The computational time of the proposed algorithm is very small and compared to other metaheuristic algorithms is ranked in the first place for the first set of instances and in the second place for the large scale vehicle routing instances.

Table 6 Running times of various heuristics and metaheuristics algorithms in golden instances

Algorithm	CPU (min)	Computer used
VRTR [23]	0.97	Athlon 1 GHz
VRP-Bilevel	3.44	Pentium III 667 MHz
GTS [46]	17.55	Pentium 200 MHz
LBTA [42]	17.81	Pentium 233 MHz
BATA [41]	18.41	Pentium 233 MHz
RTR [17]	37.15	Pentium 100 MHz
BoneRoute [43]	42.05	Pentium 400 MHz
stand-SEPAS [40]	45.48	Pentium II 400 MHz
aver-D-Ants [34]	49.33	Pentium 400 MHz
LS [9]	49.89	not mentioned
LT [9]	60.88	not mentioned
st-Prins [32]	66.6	Pentium 1,000 MHz
Xu and Kelly [47]	1825.59	DEC ALPHA Workstation

References

1. Baker, B.M., Ayechew, M.A.: A genetic algorithm for the vehicle routing problem. Comput. Oper. Res. **30**(5), 787–800 (2003)
2. Barbarosoglu, G., Ozgur, D.: A tabu search algorithm for the vehicle routing problem. Comput. Oper. Res. **26**, 255–270 (1999)
3. Berger, J., Mohamed, B.: A hybrid genetic algorithm for the capacitated vehicle routing problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 646–656. Chicago (2003)
4. Bodin, L., Golden, B.: Classification in vehicle routing and scheduling. Networks **11**, 97–108 (1981)
5. Bodin, L., Golden, B., Assad, A., Ball, M.: The state of the art in the routing and scheduling of vehicles and crews. Comput. and Oper. Res. **10**, 63–212 (1983)
6. Bullnheimer, B., Hartl, P.F., Strauss, C.: An improved ant system algorithm for the vehicle routing problem. Ann. Oper. Res. **89**, 319–328 (1999)
7. Christofides, N.: Vehicle routing. In: Lawer, E.L., Lenstra, J.K., Rinnoy Kan, A.H.G., Shmoys, D.B. (ed.) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, pp. 431–448. New York (1985)
8. Cordeau, J.F., Gendreau, M., Laporte, G., Potvin, J.Y., Semet, F.: A guide to vehicle routing heuristics. J. Oper. Res. Soc. **53**, 512–522 (2002)
9. Coy, S.P., Golden, B.L., Runger, G.C., Wasil, E.A.: Using experimental design to effective parameter settings for heuristics. J. Heuristics **7**(1), 77–97 (2001)
10. Fisher, M.L., Jaikumar, R.: A generalized assignment heuristic for vehicle routing. In: Golden, B., Bodin, L.(ed.) Proceedings of the International Workshop on Current and Future Directions in the Routing and Scheduling of Vehicles and Crews, New York (1979) Wiley, pp. 109–124.
11. Fisher, M.L.: 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, pp. 1–33 (1995)
12. Garfinkel, R., Nemhauser, G.: Integer Programming. J Wiley, New York (1972)
13. Gendreau, M., Hertz, A., Laporte, G.: A tabu search heuristic for the vehicle routing problem. Manage. Sci. **40**, 1276–1290 (1994)
14. Gendreau, M., Laporte, G., Potvin, J-Y.: Vehicle routing: modern heuristics. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, (1997) Wiley, Chichester, pp. 311–336.
15. Gendreau, M., Laporte, G., Potvin, J.Y.: Metaheuristics for the capacitated VRP. In: Toth, P., Vigo. D. The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications, Philadelphia, MA SIAM, pp.129–154.
16. Golden, B.L., Assad, A.A.: Vehicle Routing: Methods and Studies. North Holland, Amsterdam (1988)
17. Golden, B.L., Wasil, E.A., Kelly, J.P., Chao, I.M.: The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic, T.G.,

Laporte, G. (eds.) Fleet Management and Logistics, pp. 33–56. Kluwer Academic Publishers, Boston (1998)
18. Hansen, P., Mladenovic, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**, 449–467 (2001)
19. Held, M., Karp, R.M.: The traveling salesman problem and minimum spanning trees. Oper. Res. **18**, 1138–1162 (1970)
20. Laporte, G., Gendreau, M., Potvin, J.-Y., Semet, F.: Classical and modern heuristics for the vehicle routing problem. Int. Trans. Oper. Res. **7**, 285–300 (2000)
21. Laporte, G., Semet, F.: Classical heuristics for the capacitated VRP. In: Toth, P., Vigo, D. (eds.) The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications, pp. 109–128. SIAM. Philadelphia, PA (2002)
22. Lawer, E.L., Lenstra, J.K., Rinnoy Kan, A.H.G., Shmoys, D.B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York (1985)
23. Li, F., Golden, B., Wasil, E.: Very large-scale vehicle routing: new test problems, algorithms and results, Comput. Oper. Res. **32**(5), 1165–1179 (2005)
24. Lin, S.: Computer solutions of the traveling salesman problem. Bell Sys. Tech. J. **44**, 2245–2269 (1965)
25. Marinakis, Y.: Vehicle Routing in Distribution Problems. Ph. D. Thesis. Department of Production Engineering and Management, Technical University of Crete, Chania, Greece (2005)
26. Marinakis, Y., Migdalas, A.: Heuristic Solutions of Vehicle Routing Problems in Supply Chain Management. In: Pardalos, P.M., Migdalas, A., Burkard, R. (eds.) Combinatorial and Global Optimization, pp. 205–236. World Scientific Publishing Co, Singapore, (2002)
27. Marinakis, Y., Migdalas, A., Pardalos P.M.: Expanding neighborhood GRASP for the traveling salesman problem. Comput. Optim. Appl. **32**, 231–257 (2005)
28. Marinakis, Y., Migdalas, A., Pardalos, P.M.: A Hybrid Genetic-GRASP algortihm using langrangean relaxation for the traveling salesman problem. J. Comb. Optim. **10**, 311–326 (2005)
29. Marinakis, Y., Migdalas, A., Pardalos, P.M.: Multiple phase neighborhood search GRASP based on Lagrangian relaxation and random backtracking Lin–Kernighan for the traveling salesman problem (submitted in Optimization Methods and Software (2006))
30. Migdalas, A., Pardalos, P.: Nonlinear bilevel problems with convex second level problem—Heuristics and descent methods In: Du, D. -Z. et al., (eds.) Operations Research and its Application, pp. 194–204. World Scientific, Singapore (1995)
31. Osman, I.H.: Metastrategy simulated annealing and tabu search algorithms for combinatorial optimization problems. Ann. Oper. Res. **41**, 421–451 (1993)
32. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. **31**, 1985–2002 (2004)
33. Reimann, M., Stummer, M., Doerner, K.: A savings based ant system for the vehicle routing problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, 1317–1326. New York (2002)
34. Reimann, M., Doerner, K., Hartl, R.F.: D-Ants: savings based ants divide and conquer the vehicle routing problem. Comput. Oper. Res. **31**(4), 563–591 (2004)
35. Rego, C.: A subpath ejection method for the vehicle routing problem. Manage Sci. **44**, 1447–1459 (1998)
36. Rego, C.: Node-ejection chains for the vehicle routing problem: sequential and parallel algorithms. Parallel Comput. **27**(3), 201–222 (2001)
37. Resende, M. G. C., Ribeiro, C. C.: Greedy Randomized Adaptive Search Procedures. In: pp. 219–249.Glover, F., Kochenberger, G. A. (eds.) Handbook of Metaheuristics, Kluwer Academic Publishers, Boston (2003)
38. Rochat, Y., Taillard, E.D.: Probabilistic diversification and intensification in local search for vehicle routing. J. Heuristics **1**, 147–167 (1995)
39. Taillard, E.D.: Parallel iterative search methods for vehicle routing problems. Networks **23**, 661–672 (1993)
40. Tarantilis, C.D.: Solving the vehicle routing problem with adaptive memory programming methodology. Comput. Oper. Res. bf 32(9), 2309–2327 (2005)
41. Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S.: A backtracking adaptive threshold accepting metaheuristic method for the Vehicle Routing Problem. Sys. Anal. Model. Simul. (SAMS) **42**(5), 631–644 (2002)
42. Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S.: A list based threshold accepting algorithm for the capacitated vehicle routing problem. Int. J. Comput. Math. **79**(5), 537–553 (2002)

43. Tarantilis, C.D., Kiranoudis, C.T.: BoneRoute: an adaptive memory-based method for effective fleet management. Ann. Oper. Res. **115**(1), 227–241 (2002)
44. Toth, P., Vigo, D.: The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications. SIAM Philadelphia, PA (2002a)
45. Toth, P., Vigo, D.: An overview of Vehicle Routing Problems. In: Toth, P., Vigo, D. (eds.) The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications. pp. 1–26.SIAM, Philadelphia, MA (2002b)
46. Toth, P., Vigo, D.: The granular tabu search (and its application to the vehicle routing problem). INFORMS J. Comput. **15**(4), 333–348 (2003)
47. Xu, J., Kelly, J.P.: A new network flow-based tabu search heuristic for the vehicle routing problem. Transportation Sci. **30**, 379–393 (1996)