Neurocomputing 109 (2013) 76–83

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

An improved hybrid algorithm for solving the generalized vehicle routing problem ☆

Petrică C. Pop [a], Oliviu Matei [b,*], Corina Pop Sitar [c]

[a] Technical University of Cluj-Napoca, North University Center of Baia Mare, Department of Mathematics and Computer Science, Baia Mare, Romania
[b] Technical University of Cluj-Napoca, North University Center of Baia Mare, Department of Electrical Engineering, Baia Mare, Romania
[c] Technical University of Cluj-Napoca, North University Center of Baia Mare, Department of Economics, Romania

ARTICLE INFO

Available online 17 October 2012

Keywords:
Generalized vehicle routing problem
Genetic algorithms
Local search
Hybrid algorithms

ABSTRACT

The generalized vehicle routing problem (GVRP) is a natural extension of the classical vehicle routing problem (VRP). In GVRP, we are given a partition of the customers into groups (clusters) and a depot and we want to design a minimum length collection of routes for the fleet of vehicles, originating and terminating at the depot and visiting exactly one customer from each group, subject to capacity restrictions. The aim of this paper is to present an efficient hybrid heuristic algorithm obtained by combining a genetic algorithm (GA) with a local–global approach to the GVRP and a powerful local search procedure. The computational experiments on several benchmark instances show that our hybrid algorithm is competitive to all of the known heuristics published to date.

1. Introduction

The *generalized vehicle routing problem* (GVRP) is defined as follows: given a weighted complete directed graph $G=(V,A)$ with $V=\{0,1,2,\ldots,n\}$ as the set of vertices, the set of arcs $A=\{(i,j)|\ i,j\in V, i\neq j\}$ and a nonnegative cost c_{ij} associated with each arc $(i,j)\in A$. The set of vertices (nodes) is partitioned into $k+1$ mutually exclusive nonempty subsets, called *clusters*, $V_0,V_1,\ldots,V_k$ (i.e. $V=V_0\cup V_1\cup\cdots\cup V_k$ and $V_l\cap V_p=\emptyset$ for all $l,p\in\{0,1,\ldots,k\}$ and $l\neq p$). The cluster V_0 has only one vertex 0, which represents the depot, and remaining n nodes belonging to the remaining k clusters represent the geographically dispersed customers. Each customer has a certain amount of demand and the total demand of each cluster can be satisfied via any of its nodes. There exists a fleet of vehicles consisting of m identical vehicles, each with a capacity Q.

The aim of the GVRP is to find the minimum total cost collection of routes originating and terminating at the depot, such that each cluster should be visited exactly once, the entering and leaving nodes of each cluster is the same and the sum of all the demands of any route does not exceed the capacity of the vehicle Q. Therefore the GVRP involves the following two related decisions:

- choosing a node subset $S\subseteq V$, such that $|S\cap V_i|=1$, for all $i=1,\ldots,k$.
- finding a minimum cost collection of routes in the subgraph of G induced by S, fulfilling the capacity constraints.

We will call such a route (i.e. a route originating and terminating at the depot and visiting exactly one node from a number of clusters and fulfilling the capacity constraints) a *generalized route*. An illustrative scheme of the GVRP and a feasible collection of routes is shown in Fig. 1.

The *generalized vehicle routing problem* (GVRP) was introduced by Ghiani and Improta [5] and the same authors proposed as well a solution procedure by transforming the GVRP into a Capacitated Arc Routing problem for which an exact algorithm and several approximate procedures are reported in the literature.

The interest in GVRP is motivated by its considerable difficulty and as well by its practical relevance. For more information about the applications of the GVRP we refer to [1,5].

Some papers have been published dealing with integer programming formulations of the GVRP: in 2003, Kara and Bektas [7] proposed an integer programming formulation with a polynomially increasing number of binary variables and constraints and more recently Bektas et al. [2] proposed four new integer linear programming formulations for the GVRP, two based on multicommodity flow and the other two based on exponential sets of inequalities and Pop et al. [13] provided two novel integer programming models and extended as well the problem for the case in which the vertices of any cluster of each tour are contiguous.

The GVRP reduces to the classical Vehicle Routing Problem (VRP) when all the clusters are singletons and to the Generalized Traveling Salesman Problem (GTSP) when $Q=\infty$. The GVRP is

☆ This is a modified version of the paper "An efficient soft computing approach to the generalized vehicle routing problem", by P.C. Pop, O. Matei and H. Valean (12) published in the proceedings of SOCO 2011.

* Corresponding author.
E-mail address: oliviu.matei@ubm.ro (O. Matei).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.03.032

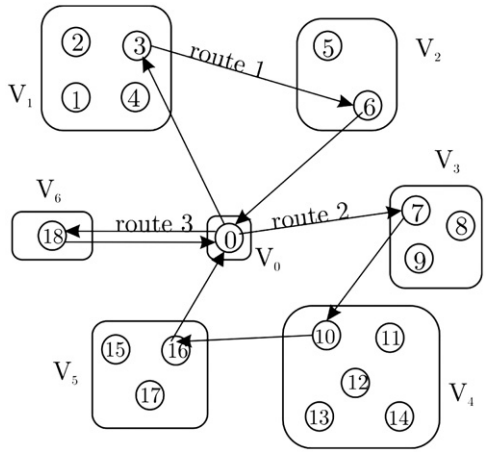

Fig. 1. An example of a feasible solution of the generalized vehicle routing problem (GVRP).

NP-hard because it includes the generalized traveling salesman problem as a special case when $Q=\infty$.

Soft computing consists of the use of computational techniques and intelligent systems for solving complex optimization problems. Several attempts to solve real-world problem can be found in the literature, see for example [3,4,15,16].

The difficulty of obtaining optimal solutions for the GVRP has led to the development of soft computing approaches: the first such algorithms were the ant colony algorithm of Pop et al. [10], the genetic algorithm of Pop et al. [11] and recently a branch-and-cut algorithm and an adaptive large neighborhood search proposed by Bekta et al. [2] an incremental tabu search heuristic described by Moccia et al. [8].

In order to solve the GVRP we propose in this paper a hybrid heuristic algorithm that combines a genetic algorithm (GA) with a local–global approach to the problem and with a powerful local search (LS) procedure. Some of the advantages of the proposed heuristic are that we reduce considerably the search space by using the GA with respect to the global graph (see Section 2) and the power and diversity of the local search heuristic (see Section 3). The obtained computational results show that our algorithm can compete with the existing algorithms for solving the GVRP.

2. The local–global approach to the GVRP

The GVRP belongs to the class of generalized combinatorial optimization problems (GCOPs), representing a natural extension of the classical combinatorial optimization problems by considering a related problem on a clustered graph, where the original problem's feasibility constraints are expressed in terms of the clusters, i.e. node sets instead of individual nodes. In the literature several GCOPs have already been considered such as the generalized minimum spanning tree problem, the generalized traveling salesman problem, the generalized vehicle routing problem, the generalized (subset) assignment problem, the generalized fixed-charge network design problem, etc.

Based on the definition of the GCOPs, a natural approach that takes advantages between them and their classical variants, is the *local–global* approach introduced by Pop [9] for the first time in the case of the generalized minimum spanning tree problem. This original approach opened new directions of research, several exact, heuristic, metaheuristic and hybrid algorithms being proposed based on it for several GCOPs.

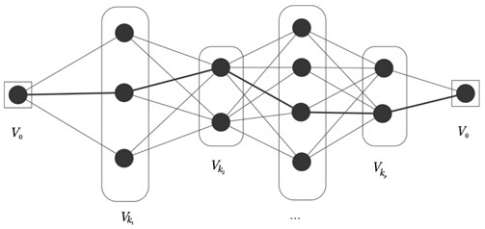

Fig. 2. Example showing a route visiting the clusters $V_0,V_{k_1},\ldots,V_{k_p}$ in the constructed layered network (LN).

The local–global approach aims at distinguishing between global connections (connections between clusters) and local connections (connections between nodes belonging to different clusters).

We denote by G' the graph obtained from G after replacing all the nodes of a cluster V_i with a supernode representing V_i, $\forall i\in\{1,\ldots,k\}$, the cluster V_0 (depot) consists already of one vertex. We will call the graph G' the *global graph*. For convenience, we identify V_i with the supernode representing it. Edges of the graph G' are defined between each pair of the graph vertices $V_0,V_1,\ldots,V_k$.

Given a solution in the global graph, i.e. a collection of r global routes of form $(V_0,V_{k_1},\ldots,V_{k_p})$ in which the clusters are visited, we want to find the best feasible route R^* (w.r.t. cost minimization), i.e. a collection of r generalized routes visiting the clusters according to the given sequence.

This can be done in polynomial time, by solving the following r shortest path problems.

For each global route $(V_0,V_{k_1},\ldots,V_{k_p})$, the best generalized route visiting the clusters according to the given sequence can be determined in polynomial time by constructing a layered network (LN) with $p+2$ layers corresponding to the clusters $V_0,V_{k_1},\ldots,V_{k_p}$ and in addition we duplicate the cluster V_0. The layered network contains all the nodes of the clusters $V_0,V_{k_1},\ldots,V_{k_p}$ plus an extra node $0'\in V_0$ and the arcs are defined as follows: there is an arc $(0,i)$ for each $i\in V_{k_1}$ with the cost c_{0i}, an arc (i,j) for each $i\in V_{k_l}$ and $j\in V_{k_{l+1}}$ $(l=1,\ldots,p-1)$ having the cost c_{ij} and an arc $(i,0')$ for each $i\in V_{k_p}$ having the cost $c_{i0'}$.

In Fig. 2 we present the constructed layered network and with bold lines a route visiting the clusters according to the given sequence $(V_0,V_{k_1},\ldots,V_{k_p})$.

We consider paths from 0 to $0'$, that visits exactly one node from each cluster $V_{k_1},\ldots,V_{k_p}$, hence it gives a feasible generalized route.

Conversely, every generalized route visiting the clusters according to the sequence $(V_0,V_{k_1},\ldots,V_{k_p})$ corresponds to a path in the layered network from $0\in V_0$ to $0'\in V_0$.

Therefore, it follows that the best (w.r.t. cost minimization) collection of routes R^* can be found by determining r shortest paths from $0\in V_0$ to the corresponding $0'\in V_0$ with the property that visits exactly one node from each of the clusters $(V_{k_1},\ldots,V_{k_p})$.

3. An improved hybrid algorithm for solving the generalized vehicle routing problem

We present in this section a hybrid algorithm for solving the GVRP obtained by combining a genetic algorithm with the local–global method described in the previous section and with a local search procedure. The proposed computational model to approach the problem is a genetic algorithm applied with respect to the global graph, reducing in this way substantially the size of the solution space.

Genetic algorithms have a predisposition for optimizing general combinatorial problems and therefore are serious candidates for solving the GVRP. However, it is known that GA are not well suited for fine-tuning structures which are close to optimal

solutions. Therefore we use local search optimization in order to refine the solutions explored by GA.

The general scheme of our hybrid heuristic is:

Step 1. *Initialization.* Construct the first generation of global solutions, i.e. collection of global routes.
Step 2. *Creation of next generation.* Use genetic operators (crossover and mutation) to produce next generation of global solutions. The parents are selected from the previous generation.
Step 3. *Producing the corresponding best GVRP generation.* For each global solution, using the procedure described in Section 2, determine its best corresponding GVRP solution.
Step 4. *Improving next generation.* Use a local search procedure to replace each of the current generation solutions by the local optimum. Eliminate duplicate solutions.
Step 5. *Termination condition.* Repeat Steps 3–5 until a stopping condition is reached.

3.1. The genetic algorithm

3.1.1. Genetic encoding

In our algorithm we used the following genetic representation of the solution domain: an individual is represented as a list of clusters

$$(V_0, V_{l_1}^{(1)}, V_{l_2}^{(1)}, \ldots, V_{l_p}^{(1)}, V_0, \ldots, V_0, V_{l_1}^{(r)}, V_{l_2}^{(r)}, \ldots, V_{l_t}^{(r)})$$

representing a collection of r global routes $V_0 - V_{l_1}^{(1)} - V_{l_2}^{(1)} - \cdots - V_{l_p}^{(1)} - V_0, \ldots$, $V_0 - V_{l_1}^{(r)} - V_{l_2}^{(r)} - \cdots - V_{l_t}^{(r)} - V_0$, where $p,t \in N$ with $1 \le p,t \le k$.

For example in the case of Fig. 1, an individual is: (1 2 0 3 4 5 0 6) and represents the collection of three global routes, which is passing through the clusters in the following order:

$(V_1\ V_2\ V_0\ V_3\ V_4\ V_5\ V_0\ V_6)$.

The values {1,…,6} represent the clusters while the depot denoted by {0} is the route splitter. Route 1 begins at the depot then visits the clusters V_1, V_2 and returns to the depot. Route 2 starts at the depot and visits the clusters $V_3 - V_4 - V_5$. Finally, in route 3 only the cluster V_6 is visited.

To the given collection of global routes corresponds several collections of generalized routes, the one represented in Fig. 1 consisting of the following nodes: $3 \in V_1$, $6 \in V_2$, $7 \in V_3$, $10 \in V_4$, $16 \in V_5$ and $18 \in V_6$. The best w.r.t. cost minimization collection of generalized routes can be determined using the approach described in Section 2.

We can see that our described representation has variable length and allows empty routes by simply placing two route splitters together without clients between them. Some routes in the chromosome may cause the vehicle to exceed its capacity. When this happens, in order to guarantee that the interpretation is always a valid candidate solution, we perform the following modification: the route that exceeds the vehicle capacity is split into several ones.

3.1.2. The fitness value

The fitness function is defined over the genetic representation and measures the quality of the represented solution. In our case, the fitness value of a feasible solution, i.e. a collection of global routes, is given by the cost of the best corresponding collection of generalized routes (w.r.t cost minimization) obtained by constructing the layered network and determined by solving a given number of shortest path problems.

3.1.3. Initial population

The construction of the initial population is of great importance to the performance of GA, since it contains most of the material the final best solution is made of. Experiments were carried out with the initial population generated randomly and with an initial population of structured solutions. In order to generate the population of structured solutions we used a Monte Carlo based method. However, from the experiments carried out it turned out that the Monte Carlo method of generating the initial population has not brought any improvements. The initial population generated randomly having the advantage that is representative from any area of the search space.

3.1.4. Genetic operators

Crossover: Two parents are selected from the population by the binary tournament method. Offspring are produced from two parent solutions using the following 2-point order crossover procedure: it creates offspring which preserve the order and position of symbols in a subsequence of one parent while preserving the relative order of the remaining symbols from the other parent. It is implemented by selecting two random cut points which define the boundaries for a series of copying operations.

The recombination of two collections of global routes requires some further explanations. First, the symbols between the cut points are copied from the first parent into the offspring. Then, starting just after the second cut-point, the symbols are copied from the second parent into the offspring, omitting any symbols that were copied from the first parent. When the end of the second parent sequence is reached, this process continues with the first symbol of the second parent until all the symbols have been copied into the offspring. The second offspring is produced by swapping round the parents and then using the same procedure.

Next we present the application of the proposed 2-point order crossover in the case of a problem consisting of eight clusters and the depot. We assume two well-structured parents chosen randomly, with the cutting points between nodes 3 and 4, respectively 6 and 7:

$P_1 = 6\ \ 8\ \ 1\ \ |\ \ 0\ \ 2\ \ 7\ \ |\ \ 0\ \ 5\ \ 4\ \ 3$
$P_2 = 8\ \ 2\ \ 1\ \ |\ \ 6\ \ 0\ \ 4\ \ |\ \ 3\ \ 5\ \ 7$

Note that the length of each individual differs according to the number of routes.

The sequences between the two cutting-points are copied into the two offspring:

$O_1 = \mathrm{x}\ \ \mathrm{x}\ \ \mathrm{x}\ \ |\ \ 0\ \ 2\ \ 7\ \ |\ \ \mathrm{x}\ \ \mathrm{x}\ \ \mathrm{x}\ \ \mathrm{x}$
$O_2 = \mathrm{x}\ \ \mathrm{x}\ \ \mathrm{x}\ \ |\ \ 6\ \ 0\ \ 4\ \ |\ \ \mathrm{x}\ \ \mathrm{x}\ \ \mathrm{x}$

The nodes of the parent P_1 are copied into the offspring O_2 if O_2 does not contain already the clusters of P_1 and therefore the offspring O_2 is

$O_2 = 8\ 1\ 2\ |\ 6\ 0\ 4\ |\ 7\ 0\ 5\ 3$

Then the nodes of the parent P_2 are copied into the offspring O_1 in the same manner. The nodes of the clusters not present in O_1 are copied into the remaining positions:

$O_1 = 8\ 1\ 6\ |\ 0\ 2\ 7\ |\ 0\ 4\ 3\ 5$

Mutation: We use in our genetic algorithm the following random mutation operator called the inter-route mutation operator which is a swap operator: it picks two random locations in the solution vector and swaps their values. Let the parent solution be (6 8 1 | 0 2 7 | 0 5 4 3) than the inter-route mutation operator picks two random clusters, for example V_8 and V_5 and swaps their values obtaining the new chromosome: (6 5 1 | 0 2 7 | 0 8 4 3).

Selection: Selection is the stage of a genetic algorithm in which individuals are chosen from a population for later breeding (crossover or mutation). The selection process is deterministic.

In our algorithm we investigated and used the properties of (μ,λ), selection, where μ parent produce λ $(\lambda > \mu)$ and only the offspring undergo selection. In other words, the lifetime of every individual is limited to only one generation. The limited life span allows to forget the inappropriate internal parameter settings. This may lead to short periods of recession, but it avoids long stagnation phases due to unadapted strategy parameters.

3.1.5. Genetic parameters

The genetic parameters are very important for the success of the algorithm, equally important as the other aspects, such as the representation of the individuals, the initial population and the genetic operators. The most important parameters are: the population size μ has been set to 10 times the number of clusters, the intermediate population size λ was chosen 10 times the size of the population: $\lambda = 10 \cdot \mu$ and the mutation probability was set at 5%. The number of epochs used in our algorithm was set to 2000.

3.2. Local search improvement

We present here five routing plan improvements used in our local search procedure. Two of them operate within the routes (the intra-route 2-opt and the 3-opt operators) and the rest operate between routes and are called relocate, exchange and cross operators. These last operators are extensions of the operators described by van Breedam [14] in the case of the classical VRP.

1. 2-*opt*: The two-opt procedure is used for improving single vehicle routes, and consists of eliminating two arcs and reconnecting the two resulting paths in a different way to obtain a new route. In Fig. 3, the two-opt operator is shown and as we can see there are different ways to reconnect the paths that yield a different tour. Among all pairs of edges whose 2-opt exchange decreases the length we choose the pair that gives the shortest tour. This procedure is then iterated until no such pair of edges is found.
2. 3-*opt*: The three-opt procedure extends the 2-opt and involves deleting three arcs in a route and reconnecting the network in all other possible ways, and then evaluating each reconnection method to find the optimum one.
3. *Relocate operator*: The relocate operator simply moves a customer visit from one route to another. It is illustrated in Fig. 4.
4. *Exchange operator*: The exchange heuristic swaps two strings of at most k vertices between two different routes. This is pictured in Fig. 5.
5. *Cross operator*: This operator includes all feasible solutions obtained by the replacement of two arc by other two. Given two different routes r_1 and r_2, an arc is deleted from each of them dividing the routes into two components that are combined such that a new route is made of the first component of r_1 and the second of r_2 and another new route is made of the first component of r_2 and the second component of r_1 (Fig. 6).

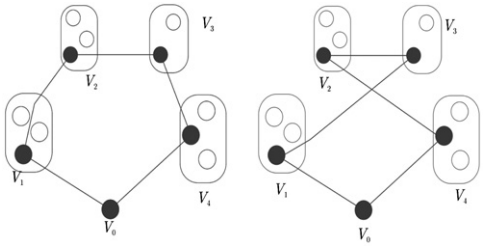

Fig. 3. An example of intra-route 2-opt exchange.

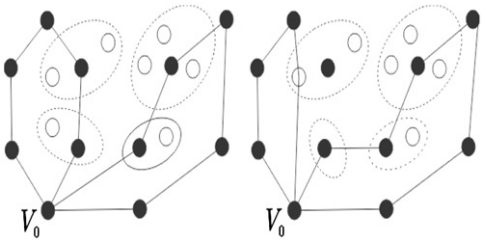

Fig. 4. An example of a possible relocation.

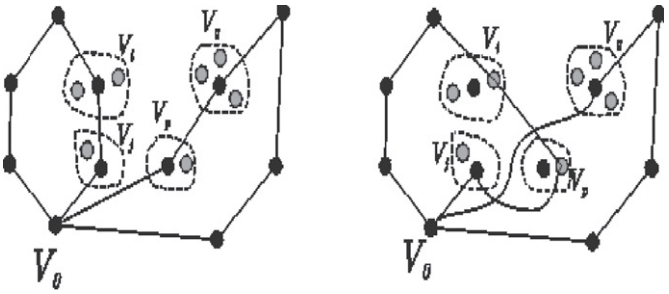

Fig. 5. An example of a possible string exchange for $k=1$.

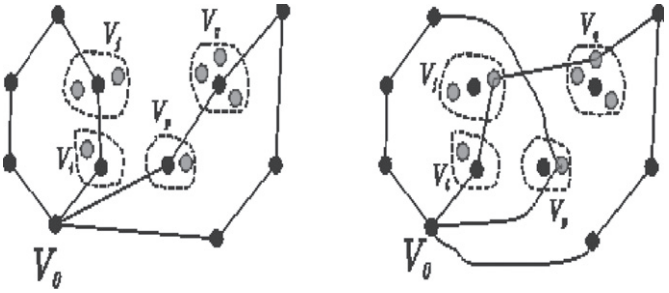

Fig. 6. An example of a possible string cross.

It is important to mention that concerning the described neighborhoods, we investigate all the possible connections of the exchanged vertices within the clusters in order to get improved routes (i.e. the nodes belonging to the marked clusters after the exchange may be different).

The local search improvement procedure applies all the described heuristics cyclically.

4. Computational results

In this section we present extensive computational results in order to assess the effectiveness of our proposed hybrid based genetic algorithm for solving the GVRP.

We conducted our computational experiments for solving the GVRP on a set of instances generated through an adaptation of the existing instances in the CVRP-library available at http://branchandcut.org/VRP/data/. The naming of the generated instances follows the general convention of the CVRP instances available online, and follows the general format $X-nY-kZ-C\Omega-V\Phi$, where X corresponds to the type of the instance, Y refers to the number

of vertices, Z corresponds to the number of vehicles in the original CVRP instance, Ω is the number of clusters and Φ is the number of vehicles in the GVRP instance. These instances were used by Bekta et al. [2] and Moccia et al. [8] in their computational experiments.

Originally the set of nodes in these problems were not divided into clusters. Fischetti et al. [6] proposed a procedure to partition the nodes of the graph into clusters, called CLUSTERING. This procedure sets the number of clusters $s=[n/\theta]$, identifies the s farthest nodes from each other and assigns each remaining node to its nearest center. Fischetti et al. [6] used $\theta=5$, but according to Bekta et al. [2] the created GVRP instances were too easy to solve.

Table 1
Computational results on small and medium instances with $\theta=2$.

Instance	*LB* [2]	*ALNS* [2]	*ITS* [8]	*HA*	100 × *HA/ALNS*	100 × *HA/ITS*
A–n32–k5–C16–V2	519	519	519	519	100	100
A–n33–k5–C17–V3	451	451	451	451	100	100
A–n36–k5–C18–V2	505	505	505	505	100	100
A–n37–k5–C19–V3	432	432	432	432	100	100
A–n38–k5–C19–V3	476	476	476	476	100	100
A–n39–k5–C20–V3	557	557	557	557	100	100
A–n39–k6–C20–V3	544	544	544	544	100	100
A–n44–k6–C22–V3	608	608	608	608	100	100
A–n45–k6–C23–V4	613	613	613	613	100	100
A–n46–k7–C23–V4	593	593	593	593	100	100
A–n48–k7–C24–V4	667	667	667	667	100	100
A–n53–k7–C27–V4	603	603	603	603	100	100
A–n54–k7–C27–V4	690	690	690	690	100	100
A–n55–k9–C28–V5	699	699	699	699	100	100
A–n60–k9–C30–V5	769	769	769	769	100	100
A–n61–k9–C31–V5	638	638	638	638	100	100
A–n62–k8–C31–V4	740	740	740	740	100	100
A–n63–k10–C32–V5	801	801	801	801	100	100
A–n64–k9–C32–V5	763	763	763	763	100	100
A–n65–k9–C33–V5	682	682	682	682	100	100
A–n69–k9–C35–V5	680	680	680	680	100	100
A–n80–k10–C40–V5	957.4	997	997	1004	100.70	100.70
B–n31–k5–C16–V3	441	441	441	441	100	100
B–n34–k5–C17–V3	472	472	472	472	100	100
B–n35–k5–C18–V3	626	626	626	626	100	100
B–n38–k6–C19–V3	451	451	451	451	100	100
B–n39–k5–C20–V3	357	357	357	357	100	100
B–n41–k6–C21–V3	481	481	481	481	100	100
B–n43–k6–C22–V3	483	483	483	483	100	100
B–n44–k7–C22–V4	540	540	540	540	100	100
B–n45–k5–C23–V3	497	497	497	497	100	100
B–n50–k7–C25–V4	449	449	449	449	100	100
B–n51–k7–C26–V4	651	651	651	651	100	100
B–n52–k7–C26–V4	450	450	450	450	100	100
B–n56–k7–C28–V4	486	486	486	486	100	100
B–n57–k7–C29–V4	751	751	751	751	100	100
B–n63–k10–C32–V5	816	816	816	816	100	100
B–n64–k9–C32–V5	509	509	509	509	100	100
B–n66–k9–C33–V5	808	808	808	808	100	100
B–n67–k10–C34–V5	673	673	673	673	100	100
B–n68–k9–C34–V5	704	704	704	704	100	100
B–n78–k10–C39–V5	803	803	803	803	100	100
P–n16–k8–C8–V5	239	239	239	239	100	100
P–n19–k2–C10–V2	147	147	147	147	100	100
P–n20–k2–C10–V2	154	154	154	154	100	100
P–n21–k2–C11–V2	160	160	160	160	100	100
P–n22–k8–C11–V5	314	314	314	314	100	100
P–n23–k8–C12–V5	312	312	312	312	100	100
P–n40–k5–C20–V3	294	294	294	294	100	100
P–n45–k5–C23–V3	337	337	337	337	100	100
P–n50–k10–C25–V5	410	410	410	410	100	100
P–n50–k7–C25–V4	353	353	353	353	100	100
P–n51–k10–C26–V6	427	427	427	427	100	100
P–n55–k10–C28–V5	415	415	415	415	100	100
P–n55–k15–C28–V8	545.3	555	565	560	100.90	99.11
P–n55–k7–C28–V4	361	361	361	361	100	100
P–n60–k10–C30–V5	433	443	443	443	100	100
P–n60–k15–C30–V8	553.9	565	565	565	100	100
P–n65–k10–C33–V5	487	487	487	487	100	100
P–n70–k10–C35–V5	485	485	485	485	100	100
P–n76–k4–C38–V2	383	383	383	383	100	100
P–n76–k5–C38–V3	405	405	405	405	100	100
P–n101–k4–C51–V2	455	455	455	455	100	100

We considered for our instances as in Bekta et al. [2], Moccia et al. [8] two clustering procedures one with $\theta = 2$ and the other one with $\theta = 3$.

However, the solution approach proposed in this paper is able to handle any cluster structure.

The testing machine was an Intel Dual-Core 1.6 GHz and 1 GB RAM. The operating system was Windows XP Professional. The algorithm was developed in Java, JDK 1.6.

In Tables 1 and 2, we summarize the results of our proposed hybrid algorithm in comparison to the adaptive large neighborhood

Table 2
Computational results on small and medium instances with $\theta = 3$.

Instance	*LB* [2]	*ALNS* [2]	*ITS* [8]	*HA*	100 × *HA/ALNS*	100 × *HA/ITS*
A–n32–k5–C11–V2	386	386	386	386	100	100
A–n33–k5–C11–V2	315	318	315	315	99.05	100
A–n36–k5–C12–V2	396	396	396	396	100	100
A–n37–k5–C13–V2	347	347	347	347	100	100
A–n38–k5–C13–V2	367	367	367	367	100	100
A–n39–k5–C13–V2	364	364	364	364	100	100
A–n44–k6–C15–V2	503	503	503	503	100	100
A–n45–k6–C15–V3	474	474	474	474	100	100
A–n46–k7–C16–V3	462	462	462	462	100	100
A–n48–k7–C16–V3	451	451	451	451	100	100
A–n53–k7–C18–V3	440	440	440	440	100	100
A–n54–k7–C18–V3	482	482	482	482	100	100
A–n55–k9–C19–V3	473	473	473	473	100	100
A–n60–k9–C20–V3	595	595	595	595	100	100
A–n61–k9–C21–V4	473	473	473	473	100	100
A–n62–k8–C21–V3	596	596	596	596	100	100
A–n63–k9–C21–V3	625.6	642	643	643	100.15	100
A–n64–k9–C22–V3	536	536	536	536	100	100
A–n65–k9–C22–V3	500	500	500	500	100	100
A–n69–k9–C23–V3	520	520	520	520	100	100
A–n80–k10–C27–V4	679.4	710	710	710	100	100
B–n31–k5–C11–V2	356	356	356	356	100	100
B–n34–k5–C12–V2	369	369	369	369	100	100
B–n35–k5–C12–V2	501	501	501	501	100	100
B–n38–k6–C13–V2	370	370	370	370	100	100
B–n39–k5–C13–V2	280	280	280	280	100	100
B–n41–k6–C14–V2	407	407	407	407	100	100
B–n43–k6–C15–V2	343	343	343	343	100	100
B–n44–k7–C15–V3	395	395	395	395	100	100
B–n45–k5–C15–V2	410	422	410	410	97.15	100
B–n50–k7–C17–V3	393	393	393	393	100	100
B–n51–k7–C17–V3	511	511	511	511	100	100
B–n52–k7–C18–V3	359	359	359	359	100	100
B–n56–k7–C19–V3	356	356	356	356	100	100
B–n57–k7–C19–V3	558	558	558	558	100	100
B–n63–k10–C21–V3	599	599	599	599	100	100
B–n64–k9–C22–V4	452	452	452	452	100	100
B–n66–k9–C22–V3	609	609	609	609	100	100
B–n67–k10–C23–V4	558	558	558	558	100	100
B–n68–k9–C23–V3	523	523	523	523	100	100
B–n78–k10–C26–V4	606	606	606	606	100	100
P–n16–k8–C6–V4	170	170	170	170	100	100
P–n19–k2–C7–V1	111	111	111	111	100	100
P–n20–k2–C7–V1	117	117	117	117	100	100
P–n21–k2–C7–V1	117	117	117	117	100	100
P–n22–k2–C8–V1	111	111	111	111	100	100
P–n22–k8–C8–V4	249	249	249	249	100	100
P–n23–k8–C8–V3	174	174	174	174	100	100
P–n40–k5–C14–V2	213	213	213	213	100	100
P–n45–k5–C15–V2	238	238	238	238	100	100
P–n50–k10–C17–V4	292	292	292	292	100	100
P–n50–k8–C17–V3	262	262	262	262	100	100
P–n51–k10–C17–V4	309	309	309	309	100	100
P–n55–k10–C19–V4	301	301	301	301	100	100
P–n55–k15–C19–V6	378	378	378	378	100	100
P–n55–k8–C19–V3	274	274	274	274	100	100
P–n60–k10–C20–V4	325	325	325	325	100	100
P–n60–k15–C20–V5	379.2	382	382	382	100	100
P–n65–k10–C22–V4	372	372	372	372	100	100
P–n70–k10–C24–V4	385	385	385	385	100	100
P–n76–k4–C26–V2	309	320	309	309	96.56	100
P–n76–k5–C26–V2	309	309	309	309	100	100
P–n101–k4–C34–V2	370	374	370	370	98.93	100

Table 3
Computational results for large-scale instances with $\theta = 2$.

Instance	*LB* [2]	*ALNS* [2]	*ITS* [8]	*HA*	100 × *HA/ALNS*	100 × *HA/ITS*
M–n101–k10–C51–V5	542	542	542	542	100	100
M–n121–k7–C61–V4	707.7	719	720	720	100.13	100
M–n151–k12–C76–V6	629.9	659	659	659	100	100
M–n200–k16–C100–V8	744.9	791	805	791	100	98.26
G–n262–k25–C131–V12	2863.5	3249	3319	3278	100.32	98.76

Table 4
Computational results for large-scale instances with $\theta = 3$.

Instance	*LB* [2]	*ALNS* [2]	*ITS* [8]	*HA*	100 × *HA/ALNS*	100 × *HA/ITS*
M–n101–k10–C34–V4	458	458	458	458	100	100
M–n121–k7–C41–V3	527	527	527	527	100	100
M–n151–k12–C51–V4	465	483	483	483	100	100
M–n200–k16–C67–V6	563.13	605	605	605	100	100
G–n262–k25–C88–V9	2102	2476	2463	2484	100.32	100.85

search [2] and the incremental tabu search [8] on 126 test small to medium instances with $\theta = 2$ and $\theta = 3$. The first column in the tables give the name of the instances, the second column provides the values of the best lower bounds in the branch-and-cut tree [2]. Next three columns contain the values of the best solutions obtained using the adaptive large neighborhood search (ALNS), the incremental tabu search (ITS) and our proposed hybrid heuristic algorithm. The last two columns present a solution quality index computed by scaling the solution value obtained using our MA to the corresponding solution obtained using the ALNS, respectively ITS heuristic.

Analyzing the results presented in Table 1, we observe that in the case of the instance $A{-}n80{-}k10{-}C40{-}V5$ our algorithm returned a solution with a gap of 0.07% from the best solution provided by the ALNS and ITS and in the case of the instance $P{-}n55{-}k15{-}C28{-}V8$ the solution returned by our algorithm exhibits a gap of 0.05% with the best known solution provided by the ALNS, but better than the solution provided by ITS and for the rest of the instances for which the optimal solution is known we solved them optimally with our algorithm.

Table 2 shows that in 62 instances out of 63 the best solution is found by our proposed hybrid algorithm. In the case of the instance $A{-}n63{-}k9{-}C21{-}V3$, the solution produced by our algorithm has a gap of equal to 0.01% to the best known solution provided by ALNS.

Summarizing the results of Tables 1 and 2, we can conclude that our hybrid heuristic algorithm is competitive in comparison to all other approaches for solving both small and medium size instances.

Next we present the results of the computational experiments on the large-scale instances generated using $\theta = 2$ and $\theta = 3$ in Tables 3 and 4, respectively.

Analyzing Tables 3 and 4, we see that overall our hybrid heuristic is competitive in comparison to ALNS and ITS heuristics as well in the case of large-size instances.

Regarding the computational times, it is difficult to make a fair comparison between algorithms, because they have been evaluated on different computers and they are implemented in different languages. The running time of our hybrid algorithm is proportional with the number of generations. From the computational experiments, it seems that 10^4 generations are enough to explore the solution space of the GVRP. Our proposed heuristic algorithm seems to be slower than ALNS and comparable with ITS. Therefore we can conclude that our approach will be appropriate when the execution speed is not critical.

5. Conclusions

In this paper, we developed an improved hybrid algorithm for solving the GVRP. Our algorithm was obtained by combining a genetic programming with a local–global approach to the problem and with a powerful local search procedure. Our local search procedure consists of five local search heuristics, their diversity and power being an important factor for solving successfully several instances of the GVRP. Another important factor is the reduction of the solution space by using the GA with respect to the global graph.

The experimental results show that our algorithm is robust and compares well with all known heuristic approaches to the problem in terms of solution quality.

Acknowledgments

This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS – UEFISCDI, project number PN-II-RU-TE-2011-3-0113. The authors are grateful to the anonymous referees for reading the manuscript very carefully and providing constructive comments which helped to improve substantially the paper.

References

[1] R. Baldacci, E. Bartolini, G. Laporte, Some applications of the generalized vehicle routing problem, J. Oper. Res. Soc. 61 (7) (2010) 1072–1077.
[2] T. Bektas, G. Erdogan, S. Ropke, Formulations and branch-and-cut algorithms for the generalized vehicle routing problem, Transp. Sci. 45 (3) (2011) 299–316.
[3] E. Corchado, A. Herrero, Neural visualization of network traffic data for intrusion detection, Appl. Soft Comput. 11 (2) (2011) 2042–2056.
[4] E. Corchado, B. Baruque, WeVoS-ViSOM: an ensemble summarization algorithm for enhanced data visualization, Neurocomputing 75 (1) (2012) 171–184.
[5] G. Ghiani, G. Improta, An efficient transformation of the generalized vehicle routing problem, Eur. J. Oper. Res. 122 (1) (2000) 11–17.
[6] M. Fischetti, J.J. Salazar, P. Toth, A branch-and-cut algorithm for the symmetric generalized traveling salesman problem, Oper. Res. 45 (3) (1997) 378–394.
[7] I. Kara, T. Bektas, Integer linear programming formulation of the generalized vehicle routing problem, in: EURO/INFORMS Joint International Meeting, Istanbul, July 06–10, Turkey, 2003.
[8] L. Moccia, J-F. Cordeau, G. Laporte, An incremental tabu search heuristic for the generalized vehicle routing problem with time windows, J. Oper. Res. Soc. 2010, http://dx.doi.org/10.1057/jors.2011.25.

[9] P.C. Pop, The Generalized Minimum Spanning Tree Problem, Ph.D. Thesis, University of Twente, The Netherlands, 2002.
[10] P.C. Pop, C. Pintea, I. Zelina, D. Dumitrescu, Solving the generalized vehicle routing problem with an acs-based algorithm, Am. Inst. Phys. 1117 (2009) 157–162.
[11] P.C. Pop, O. Matei, C. Pop Sitar, C. Chira, A genetic algorithm for solving the generalized vehicle routing problem, in: Lecture Notes in Artificial Intelligence, vol. 6077, 2010, pp. 119–126.
[12] P.C. Pop, O. Matei, H. Valean, An efficient soft computing approach to the generalized vehicle routing problem, Adv. Intelligent Soft Comput. 87 (2011) 281–289.
[13] P.C. Pop, I. Kara, A. Horvat Marc, New mathematical models of the generalized vehicle routing problem and extensions, Appl. Math. Modelling 36 (1) (2012) 97–107.
[14] A. van Breedam, An Analysis of the Behavior of the Heuristics of the Vehicle Routing Problem for a Selection of Problems, with Vehicle-related, Customer-related and Time-related Constraints, Ph.D. Dissertation, University of Antwerp, Belgium, 1994.
[15] J. Sedano, L. Curiel, E. Corchado, E. de la Cal, J.R. Villar, A soft computing based method for detecting lifetime building thermal insulation failures, Integrated Comput. Aided Eng. 17 (2) (2010) 103–115.
[16] T. Wilk, M. Wozniak, Soft computing methods applied to combination of one-class classifiers, Neurocomputing 75 (1) (2012) 185–193.

Petrica Claudiu Pop received B.S. degree in mathematics from the Babes-Bolyai University of Cluj-Napoca, Romania, the M.S. degree in operations research at the University of Twente, the Netherlands, and the Ph.D. degree in operations research at the same university. His research interests include combinatorial optimization, mathematical modeling, artificial intelligence and mathematical programming. Several research stays have taken him to Italy, UK, Greece, France, the Netherlands, Austria and Canada. He serves as an Associate Professor of informatics and mathematics at the North University of Baia Mare, Romania.

Oliviu Matei received his B.Sc.Eng. in computer science from the Technical University of Cluj Napoca, Romania, and his M.Sc. in artificial intelligence from Vrije Universiteit of Amsterdam, the Netherlands. He is a Ph.D. student in computer science at the Technical University of Cluj-Napoca, Romania. His main interests are evolutionary computing, autonomous robots, machine learning and data mining.

Corina Pop Sitar received the M.Sc. and Ph.D. degrees in economics from the Babes-Bolyai University, Cluj Napoca, Romania, in 1998 and 2007. Previously, during 1998 and 2002 she was a research fellow at Twente University, the Netherlands, and her main research fields were supply chain management and operation research. Currently, she is a lecturer at North University of Baia Mare, Romania. Her research interests are E-procurement, supply chain management, combinatorial optimization and applied mathematics.