INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 21 (2014) 153–175
DOI: 10.1111/itor.12041

Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size

H. Murat Afsar, Christian Prins and Andréa Cynthia Santos

ICD-LOSI, UMR CNRS STMR, Université de Technologie de Troyes, 12, rue Marie Curie, CS 42060, 10004, Troyes, France

E-mail: murat.afsar@utt.fr [Afsar]; christian.prins@utt.fr [Prins]; andrea.duhamel@utt.fr [Santos]

Received 16 May 2012; received in revised form 20 March 2013; accepted 20 June 2013

Abstract

The generalized vehicle routing problem with flexible fleet size (GVRP-flex) extends the classical capacitated vehicle routing problem (CVRP) by partitioning the set of required nodes into clusters and has interesting applications such as humanitarian logistics. The problem aims at minimizing the total cost for a set of routes, such that each cluster is visited exactly once and its total demand is delivered to one of its nodes. An exact method based on column generation (CG) and two metaheuristics derived from iterated local search are proposed for the case with flexible fleet size. On five sets of benchmarks, including a new one, the CG approach often provides good upper and lower bounds, whereas the metaheuristics find, in a few seconds, solutions with small optimality gaps.

Keywords: generalized vehicle routing problem; column generation; splitting procedure; iterated local search; humanitarian logistics; disaster response

1. Introduction

Natural disasters often cause irreparable and dramatic losses to populations and the environment. An efficient organization of relief operations is essential to mitigate these effects and reduce the risks of overmortality and propagation. Even when the relief agencies and their donors can provide abundant aid, its distribution can be seriously impaired if the logistic aspects are neglected. The recent large-scale disasters such as the 2010 earthquake in Haiti or the 2004 tsunami in Indonesia have highlighted the consequences of logistic deficiencies. For instance, only 5% of the ruins were cleared away in the capital of Haiti one year after the earthquake. A more efficient logistic management in humanitarian disasters is then a promising application field for operations research.

Disaster relief operations require efficient evacuation or delivery of resources such as medical staff or equipment to damaged sites, spread over a vast area. The road network is often so damaged that

the first wave of relief is performed by planes. To save time, the resources are dropped at one of the airports that is still operational in each region, the local authorities dealing with the distribution in the rest of the region. The regions to be visited are not necessarily administrative, they can be defined by other criteria such as proximity, easy distribution from the selected airport (e.g., a connected component of the residual road network), geographical restrictions (e.g., a valley surrounded by high mountains), etc. This kind of distribution can also be applied to archipelagos, if one delivery harbor selected for each island.

These situations can be modeled as a generalized vehicle routing problem with flexible fleet size (GVRP-flex), an extension of the capacitated vehicle routing problem (CVRP) in which the set of customers is partitioned into clusters. From a depot with a fleet of vehicles of known capacity, the problem aims at minimizing the total distance for a set of routes, such that each cluster is visited exactly once, the total demand of a cluster is delivered to one of its required node, and vehicle capacity is respected.

This work is inspired by humanitarian logistics but this is not a case study. Our concern is to propose several strategies to solve a core problem, the GVRP-flex. The objective is not only to develop software components for humanitarian logistics, but also to have efficient solution methods for the GVRP-flex. After a literature review in Section 2 the problem and basic notations are introduced in Section 3 An exact method based on column generation is proposed in Section 4 while Section 5 reports a splitting procedure called by iterated local search (ILS) based metaheuristics described in Section 6. This splitting procedure is used to deduce a feasible GVRP-flex solution from a giant tour. Computational results are presented in Section 7, followed by concluding remarks in Section 8.

2. Literature review

2.1. Vehicle routing in humanitarian logistics

The location and transportation issues in disaster management have attracted many operational research (OR) specialists, impulsing a stream of research called "emergency logistics" or "humanitarian logistics." The reader may refer to three good surveys as entry points. Altay and Green (2006) review the literature on disaster operations management and split the activities into four phases: mitigation, preparedness, response, and recovery. More recently, de la Torre et al. (2012) consider several families of problems and OR models: allocation policies, needs assessment, uncertainty in demands and supplies, vehicles, and routes. Caunhye et al. (2012) distinguish between predisaster operations (short-notice evacuation, stock prepositioning, facility location for shelters, stores, and medical centers) and postdisaster activities (relief distribution, casualty transportation).

The subset of transportation problems in disaster management is not limited to vehicle routing for distributing relief. For instance, Saadatseresht et al. (2009) propose a genetic algorithm to solve an evacuation planning problem. Yi and Özdamar (2007) study a dynamic problem combining evacuation of injured people and deliveries, modeled by flow techniques. In fact, the three surveys cited before show that vehicle routing problems that take place in the response phase still receive little attention.

As we have not found publications on the applications of GVRP or GVRP-flex to emergency logistics, we give a few examples of vehicle routing problems already studied in this field. There are two main groups of existing works.

The first group consists of variants of the CVRP, but with different objective functions. For instance, Campbell et al. (2008) minimize the average arrival time (min-avg VRP) or the latest arrival time (min-max VRP) at customers. Using insertion heuristics and a comparison with CVRP solutions, they show that these two approaches result in faster delivery, but at the expense of a higher total transportation cost. In Ngueveu et al. (2010), the authors solve the min-avg VRP (that they call cumulative VRP) using a memetic algorithm, that is, a genetic algorithm hybridized with a local search procedure. Ribeiro and Laporte (2012) improve their results using an adaptive large neighborhood search (LNS). In the same vein, Huang et al. (2012) formulate different performance metrics taking into account efficacy (speed or sufficiency of deliveries), efficiency (total travel time for selected routes), and equity (e.g., the spread of service levels across nodes).

The second group copes with more complex and specialized vehicle routing problems, which can differ considerably according to the kind of disaster considered and the operations to be performed. For instance, Van Hentenryck et al. (2011) worked on restoring water, electricity, and gas infrastructures after hurricanes. The general objective is to repair, as fast as possible, these structures to minimize the affected population, which includes not only vehicle routing for the last mile but also strategic decisions such as stockpiling of power system supplies (Coffrin et al., 2011). The results from this research have provided recommendations for the Homeland Security Department in the United States.

Barbarosoglu et al. (2002) propose an interactive approach to route helicopters, with complications such as refueling operations and pilots with specialized skills. Results are presented for a simplified scenario inspired from a real disaster in Turkey. A warehouse location-routing problem with three objectives submitted by the Austrian Red Cross is described by Rath and Gutjhar (2011): warehouses must be selected among potential sites and routes must be constructed from the open sites. Lin et al. (2011) describe a tactical vehicle routing problem over a multiperiod horizon, with a limited fleet, soft time windows, several commodities with urgency levels, and penalties for late deliveries. The objective combines the total travel time and sum of penalties. Berkoune et al. (2012) consider several products too, but their version over a single period involves several distribution centers (DC), one heterogeneous fleet per DC, and loading/unloading times for each vehicle type. Vehicles may do several trips, subject to a maximum working time.

2.2. GVRP research

The literature on the GVRP is even more scarce than its uncapacitated version, the generalized traveling salesman problem (GTSP). The GVRP was introduced in 2000 by Ghiani and Improta (2000). These authors present a transformation of the GVRP into the capacitated arc routing problem (CARP) and solve one instance with 50 nodes, 24 effective clusters, and 1 artificial cluster for the depot, via a metaheuristic called CARPET. Baldacci et al. (2010) show that a number of classical problems can be modeled as a GVRP, for example, the traveling salesman problem with profits, the periodic VRP, and the VRP with selective backhauls. However, no computational experiments are reported.

Concerning exact methods, Kara and Bektas (2003) published integer linear formulations for the GVRP and the GTSP. Solving directly by means of CPLEX, these authors report results for several GTSP instances and the 24-cluster GVRP used in Ghiani and Improta (2000). The latter is solved to optimality but in 17,600 seconds on a 1.1 GHz PC. Quite recently, Bektas et al. (2011) designed a branch-and-cut procedure and a LNS for the GVRP with limited fleet.

Regarding heuristics, Pop et al. (2011) describe greedy algorithms (inspired by the nearest neighbor principle and the Clarke–Wright savings heuristic for the CVRP) and local search procedures, but without computational experiments. Two other papers provide numerical results, but for different versions of the GVRP: Moccia et al. (2011) develop a tabu search for the GVRP with time windows and limited fleet, a problem inspired by the location of stops for school bus routing, and test it on instances with and without time windows. Finally, Pop et al. (2010) design a genetic algorithm for the split GVRP, in which the total demand of a cluster can be supplied by more than one vehicle.

This short review shows that very few solution methods, either exact or heuristic, are available for the GVRP, explaining perhaps the scarcity of possible applications to relief distribution in the literature on emergency logistics. This motivated us to design algorithms able to tackle large GVRP instances.

3. Problem definition and notation

The GVRP-flex is defined on a complete undirected graph $G = (V, E)$. The node-set V contains one depot-node 0 and n demand nodes, indexed from 1 to n. A fleet of homogeneous vehicles with capacity Q is based on the depot. The fleet is here virtually unlimited, that is, the number of vehicles used is a decision variable in our algorithms. A demand d_i is associated with each demand node i. In the edge-set E, edge $[i, j]$ models a shortest path linking nodes i and j in the real network, with a precomputed cost c_{ij} (distance or travel time). Thus, the triangular inequality holds.

The set V is partitioned into m clusters, that is, $V = C_0 \cup C_1 \cup \ldots \cup C_m$ and $C_i \cap C_j = \emptyset$, $\forall i, j \leq m, i \neq j$. C_0 is an artificial cluster reduced to the depot while the other clusters, called "effective clusters," contain demand nodes. $|C_j|$, $\forall j = 1, 2, \ldots, m$, denotes the number of nodes belonging to cluster C_j. The total demand of cluster C_j is denoted as $q_j = \sum_{i \in C_j} d_i$. To ensure problem feasibility, $q_j \leq Q$ for each cluster C_j, $j = 1, 2, \ldots, m$.

The GVRP-flex consists in determining a set of routes with minimum total distance. Each route is a cycle performed by one vehicle that leaves the depot, visits a subset of clusters, and returns to the depot. Each cluster is visited by exactly one vehicle and its total demand is delivered to one of its nodes, called the "active node." The total demand satisfied by a route must fit vehicle capacity.

Figure 1 illustrates the optimal solution for 12-24-ST (where ST stands for small trips), one instance with 12 effective clusters and 24 nodes used in Section 7 for computational evaluations. The dotted lines are cluster borders. The black points indicate the nodes in each cluster while the small black square corresponds to the depot. In this example, five vehicles are used to serve all clusters.

Note that the GVRP-flex contains two special cases, known to be NP-hard: the CVRP corresponds to clusters containing one node each, while the GTSP is the single-vehicle or uncapacitated version. Thus, the GVRP-flex is also NP-hard.

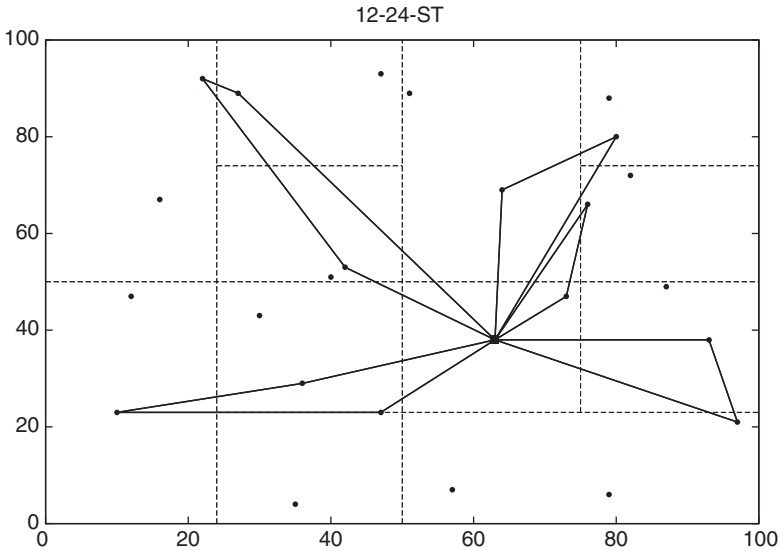

Fig. 1. Example of GVRP-flex instance with an optimal solution.

4. Column generation method

To be able to solve instances of reasonable size, the GVRP-flex is reformulated in this section using Dantzig–Wolfe decomposition, with binary variables associated with the choice of routes. As the number of possible routes is extremely large, this problem is solved by a column generation (CG) approach embedded in a branch-and-bound tree (branch-and-price, B&P). The CG method is well known and has proved its effectiveness in many routing problems, for example, see Cullen et al. (1981), Desrosiers et al. (1984), Dror and Langevin (2000). We formulate the master problem as a set-partitioning problem (SPP) and the CG subproblem as a shortest path problem with resource constraints, and we describe the branching procedure.

4.1. Set-partitioning model

Let P denote the set of all feasible routes. Each route $p \in P$ can be represented by a cost c_p and an m-bit vector γ^p whose each element γ_i^p is equal to 1 if the route visits one demand node of cluster C_i. The cost of a route p is the sum of the costs of the arcs used by this route. We define binary variables λ_p taking the value 1 if and only if feasible route p is selected. The GVRP-flex can be modeled as a SPP, in which the goal is to select a subset of routes with minimum total cost, such that each cluster is serviced once. As SPP is NP-hard and the number of feasible routes extremely large, CG is used to solve the relaxed SPP and get a good lower bound, starting with a small subset of routes P'. The following formulation corresponds to the restricted master problem (RMP). The objective function (1) to be minimized is the total length of selected routes while constraints

(2) ensure that each cluster is serviced once.

$$\min \sum_{p \in P'} c_p \lambda_p, \tag{1}$$

subject to

$$\Pi_i \longrightarrow \sum_{p \in P'} \gamma_i^p \lambda_p = 1 \quad \forall \quad i = 1, 2, \ldots, m, \tag{2}$$

$$\lambda_p \geq 0 \quad \forall p \in P'. \tag{3}$$

Each iteration consists in solving a pricing subproblem to add routes to P'. The dual variables Π_i associated with constraints (2) are used to calculate the reduced cost of the routes in the subproblem. Note that Π_i is the dual value of cluster C_i: as only one node is visited to service the entire cluster, all nodes in the same cluster have the same dual value. The feasible routes with negative reduced cost discovered by the subproblem are added to the RMP. The same steps are repeated with the new dual variables, until the pricing subproblem finds no more feasible routes with negative reduced cost.

4.2. Elementary shortest path problem with resource constraints

The aim of the pricing subproblem is to find good candidates to enter the basis of the master problem (1)–(3). A route p is feasible if and only if

- it begins and ends at the depot (node 0);
- the total demand of serviced clusters does not exceed vehicle capacity Q;
- each cluster is serviced at most once by the route.

Hence, the reduced cost of the route p

$$\bar{c}_p = c_p - \sum_{i=1}^{m} \gamma_i^p \Pi_i. \tag{4}$$

The pricing subproblem is an elementary shortest path problem with resource constraints solved using a label-setting algorithm (Feillet et al., 2004). A label $l_p = (\bar{c}_p, Q_p, v_p, K_p^1, \ldots, K_p^m)$ is associated to every partial route p, where $\bar{c}_p$ is the reduced cost, Q_p denotes the residual vehicle capacity ($Q_p = Q - \sum_i \gamma_i^p \times q_i$), v_p is the last node visited, and K_p^i, $1 \leq i \leq m$ is a binary indicator equal to 1 if and only if cluster C_i is visited by the route. In other words, a partial route is characterized by its reduced cost and its resource consumptions (vehicle capacity and clusters visited). When multiple routes lead to the same node, they are compared and dominated labels are discarded. Only the nondominated labels are extended by the nodes of unexplored clusters. Label l_p dominates (weakly) label $l_{p'}$ if and only if it visits at least the same clusters without being more expensive:

- $v_p = v_{p'}$,
- $\bar{c}_p \leq \bar{c}_{p'}$,
- $Q_p \geq Q_{p'}$,
- $K_p^i \geq K_{p'}^i \quad 1 \leq i \leq m$.

4.3. B&P algorithm

A B&P algorithm is a branch-and-bound procedure in which CG is used to provide good lower bounds at each node of the search tree. For the incumbent node, our branching procedure determines the arc $a = (v, v')$ on which the flow $\kappa_a = \sum_{p|a \in p} \lambda_p$ is fractional but closest to 1. Two child nodes are generated. In the left child, arc $a = (v, v')$ is made mandatory, that is, all outgoing arcs of node v and all incoming arcs of v' (except a) are removed. In the right child, arc a is forbidden, that is, it is removed from the network. The tree is explored in a depth-first manner. The left node is examined first to discover solutions more quickly, tracing a branch that stops when all arcs have integer flows (new feasible solution) or when the node can be pruned. Then a backtrack occurs to treat the right child of the parent node.

In our tests, the B&P is stopped after 1 hour. Four cases corresponding to growing instance sizes are possible at the end. In the best case, the tree is completely explored and one exact solution is obtained. In the second case, the tree is only partially explored and the B&P turns into a heuristic method that returns distinct lower and upper bounds. In the third case, no branching occurs: the relaxed LP is completely solved by CG at the root node and we always obtained one feasible solution by solving the same LP but with integer variables, using CPLEX. The fourth and last case happens on the largest instances: the initial LP cannot be solved in 1 hour.

5. Splitting procedure

5.1. Methods based on giant tours for CVRP and GTSP

Route-first cluster-second heuristics for vehicle routing problems consist in relaxing vehicle capacity, computing a Travelling Salesman Problem (TSP) tour that visits each customer exactly once (called "giant tour"), and splitting it into capacity-feasible trips. Beasley (1983) showed that an optimal CVRP solution (subject to the sequence) can be deduced from a given giant tour T using a splitting procedure based on an acyclic auxiliary graph $H = (X, A)$. The node-set X contains one dummy node 0 and n nodes indexed from 1 to n for the customers. If T is given as an ordered list of the n customers, any feasible trip (subsequence of customers) $(T_i, T_{i+1}, \ldots, T_j)$ is modeled in the arc-set A by one arc $(i-1, j)$, weighted by the trip cost. The optimal splitting can be obtained by computing a least-cost path from nodes 0 to n in H, the arcs along this path giving the routes of the CVRP solution. This can be done in $O(nb)$, where b is the average number of customers in feasible subsequences.

These ideas have lead to efficient genetic algorithms for the CVRP (Prins, 2004), in which chromosomes encoded as giant tours are evaluated by the splitting procedure. A similar idea was used by Bontoux et al. (2010) in a memetic algorithm for the GTSP. These authors encode a GTSP

solution as an ordering T of the m effective clusters and consider an auxiliary graph F with $m+2$ node layers. The first and last layers contain only the depot-node, while the others correspond to the customers in $T_1, T_2, \ldots, T_m$. One arc links any two customers pertaining to two successive clusters. The optimal GTSP solution for the given ordering corresponds to a shortest path linking the two depot copies in F.

5.2. Generalization for the GVRP-flex

Our goal is to design a polynomial splitting procedure for the GVRP-flex, named *Split*, called in the ILS-based metaheuristics of Section 6. Let $T = (T_1, T_2, \ldots, T_m)$ be an ordering (giant tour) of the m clusters. We build a weighted auxiliary graph $H = (X, A)$ with nodes 0 to n. Any subsequence of clusters $(T_i, T_{i+1}, \ldots, T_j)$, which can be visited by a feasible GVRP-flex route (total demand compatible with vehicle capacity), is modeled by one arc $(i-1, j)$ in A. The weight of this arc is computed using the method of Bontoux et al. (2010), that is, we compute a shortest path in another auxiliary graph F with $j-i+3$ node layers: one layer reduced to the depot, $j-i+1$ layers corresponding to the customers of clusters $T_i, T_{i+1}, \ldots, T_j$, and one final layer with the depot.

Figure 2 gives a small example for four clusters and vehicle capacity $Q = 10$. The left part shows the cluster ordering, with the Bontoux et al. (2010) method applied to subsequence (T_2, T_3, T_4). The auxiliary graph F is here the subgraph induced by the required nodes of T_2, T_3, and T_4. The best route to visit these three clusters (bold segments) is (0, 2, 3, 8, 0), with cost 17. The lower part illustrates the auxiliary graph H, in which each possible route is represented by one arc weighted by the route cost, for example, the best route for (T_2, T_3, T_4) is represented by arc (1,4). The shortest path in H is composed of the two thick arcs, with cost 35. The right part of the figure shows the resulting GVRP-flex solution with two trips.

Algorithm 1 provides an efficient implementation of *Split*, in which no auxiliary graph is generated explicitly. It is based on two nested loops (lines 4 and 7) that browse all feasible subsequences of clusters $(T_i, T_{i+1}, \ldots, T_j)$, that is, each arc $(i-1, j)$ of the implicit auxiliary graph H. For each subsequence of clusters, the total demand is computed in "load" while the minimum cost of the route is calculated by a procedure *Compute_cost*. Label V_j represents the cost of a shortest path from nodes 0 to j in the implicit auxiliary graph H (value on top of each node in Fig. 2). Whenever a better path is found to reach node j, V_j is updated, P_j stores the predecessor of j on this path, and a *Store_route* procedure records the last route of the path (route associated with the current subsequence).

Compute_cost is described in Algorithm 2. The trick for a low complexity is to deduce the best route for the input subsequence of clusters $(T_i, T_{i+1}, \ldots, T_j)$ from intermediate results computed in the previous call for $(T_i, T_{i+1}, \ldots, T_{j-1})$. To do so, a label W_u is calculated for each node contained in the clusters of the subsequence: if u belongs to cluster T_k, W_u is the node of a shortest path that leaves the depots, visits one node in clusters T_i to T_{k-1}, and ends at node $u \in T_k$. This path corresponds to a route, but without the return to the depot. The predecessor of u on this route is stored in B_u. The vectors W and B are stored as global variables to be preserved between two calls.

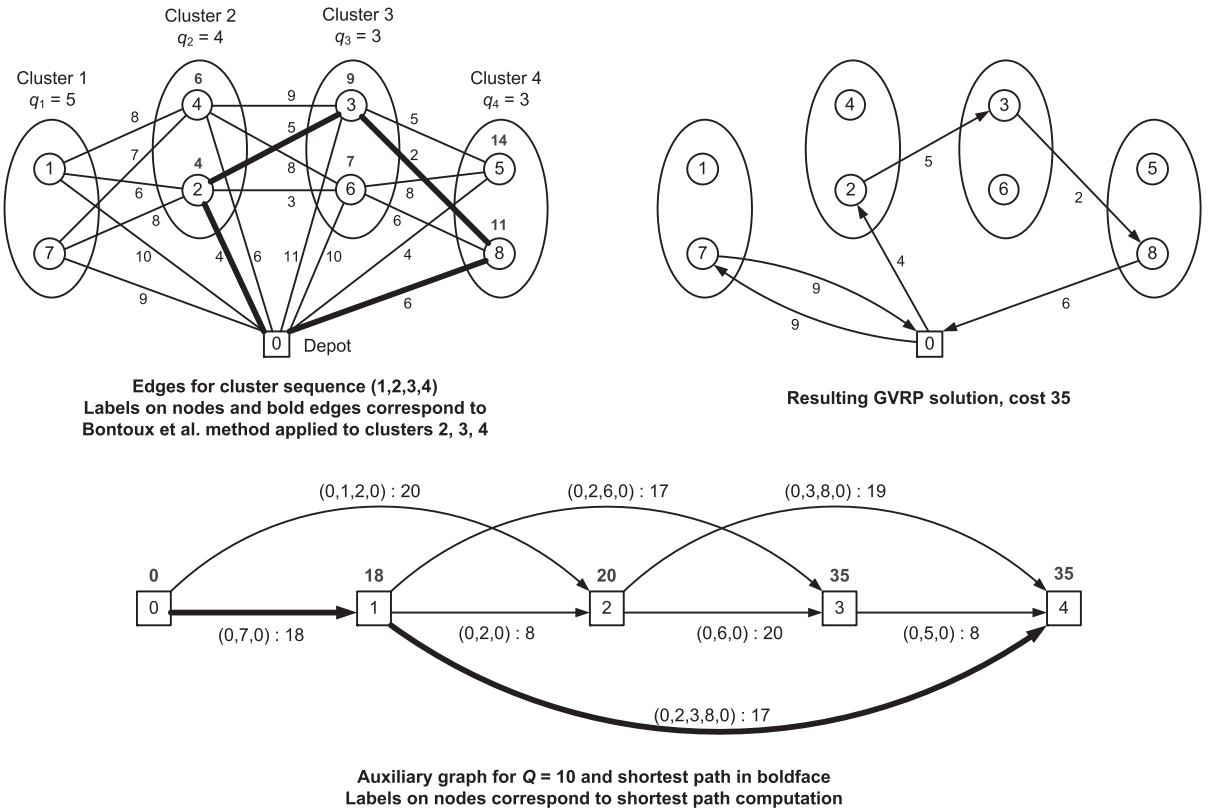

Fig. 2. Example for the splitting procedure.

For each node $u \in T_j$, the best route to u is arc $(0, u)$ if $j = i$ (line 3). Otherwise (lines 7–13), the procedure adds arc (v, u) to the routes ending at each node $v \in T_{j-1}$ and updates label W_u when improved. Lines 14–17 add arc $(u, 0)$ to return to the depot and memorize the best route as a pair $(cost, y)$, y being the last demand node. These results are necessary for tracing the route back.

Store_route (Algorithm 3) records the best route for the subsequence of clusters $(T_i, T_{i+1}, \ldots, T_j)$. Using the last node u and the predecessors stored in B (computed by *Compute_cost*), the route is traced backwards and its demand nodes are stored in row j of an $m \times m$ matrix S of demand nodes.

Finally, using Algorithm 4, the GVRP-flex solution can be extracted from the shortest path computed in graph H. The algorithm begins at the last node (cluster) m and backtracks using predecessors P_j. The solution is encoded as a list containing the customers of successive trips, separated by depot copies. For each node j of H, the trip corresponding to the arc ending at j on the shortest path has been stored in S by *Store_route*.

Note that *Split* gives an optimal GVRP-flex solution, subject to the order defined by the giant tour. It is also easy to show that there exist optimal giant tours, that is, tours yielding an optimal GVRP-flex solution after splitting. Hence, it is possible to design metaheuristics that search the space of giant tours, like the ones presented in the next section.

Algorithm 1: General structure of *Split* for one giant tour T

```
1  V(0) ← 0;
2  P(0) ← 0;
3  for i ← 1 to m do V_i ← ∞;
4  for i ← 1 to m do
5  |  j ← i;
6  |  load ← 0;
7  |  repeat
8  |  |  load ← load + q(T(j));
9  |  |  if load ≤ Q then
10 |  |  |  Compute_cost (T, i, j, cost, y);
11 |  |  |  if V(i − 1) + cost < V(j) then
12 |  |  |  |  V(j) ← V(i − 1) + cost;
13 |  |  |  |  P(j) ← i − 1;
14 |  |  |  |  Store_route (y, S, j);
15 |  |  |  end
16 |  |  |  j ← j + 1;
17 |  |  end
18 |  until (load > Q) or (j > m);
19 end
20 return (V_m, P, B)
```

If b is the average number of clusters per route (average out-degree of the nodes in H), it is possible to show that *Split* runs in $O(n^2 b/m)$. If $m = n$ (one customer per cluster), we find the $O(nb)$ complexity already mentioned for the CVRP in Section 5.1.

6. ILS-based metaheuristics

6.1. Principles and general structure

ILS is a simple but effective metaheuristic, which has been successfully applied to a number of optimization problems (Lourenço et al., 2002). Starting from one initial local optimum S, each iteration takes a copy of S, applies a perturbation procedure, and improves the perturbed copy using a local search procedure. S is updated if the resulting child-solution is better. As ILS has produced high-quality results for vehicle routing problems (Prins, 2009; Nguyen et al., 2012), we propose an ILS-based metaheuristic dedicated to the GVRP-flex, with additional components such as a restart procedure. We refer to this version as the multi-start ILS (MS-ILS) heuristic. We also propose a relaxed version (R-ILS), in which the incumbent solution S is updated at each iteration, even if the child is not better.

The key feature of our MS-ILS is to alternate between the space of giant tours and the space of GVRP solutions. The internal ILS begins with one pair (S, T), where S is a GVRP solution

Algorithm 2 : Algorithm for *Compute_cost*

```
cost ← ∞;
for each u ∈ T(j) do
    if j = i then
        W_u ← c_0u;
        B_u ← 0;
    else
        W_u ← ∞;
        for each v ∈ T_{j−1} do
            if W_v + c_vu < W_u then
                W_u ← W_v + c_vu;
                B_u ← v;
            end
        end
        if W_u + c_u0 < cost then
            cost ← W_u + c_u0;
            y ← u;
        end
    end
end
```

Algorithm 3 : Algorithm for *Store_route*

```
u ← j − i + 2;
k ← y;
repeat
    u ← u − 1;
    S_ju ← k;
    k ← B_k;
until k = 0;
```

Algorithm 4 : Algorithm to extract the GVRP-flex solution

```
L ← ∅;
j ← m;
repeat
    i ← P_j + 1;
    insert at the beginning of L the route (0, S_j1, S_j2, ..., S_{j,j−i+1}, 0);
    j ← P_j;
until j = 0;
```

computed via a heuristic and T, the giant tour obtained by concatenating its routes. Then, each ILS iteration performs four steps: (I) a random perturbation is applied to a copy T' of T; (II) T' is converted into a GVRP solution S' via *Split*; (III) S' is improved using a local search procedure; (IV) if S' outperforms S, it replaces S and its routes are concatenated to get a giant tour T, giving the pair (S, T) for the next iteration. The ILS is embedded in a main loop that restarts it from several initial solutions.

The process is detailed in Algorithm 5, before presenting the components. The input parameters are the maximum number of restarts (*nb_r*) and, for each internal ILS, the maximum number of restarts (*nb_ils*) and the maximum number of iterations without improving the best solution S (*nb_f*). S^* denotes the global best solution. As 90% of the running time is spent in the calls to the local search, we preferred to define a "computing budget" $budget = nb\_r \times nb\_ils$ (line 2) and to stop the internal ILS and the main loop after *budget* calls to the local search, instead of doing a fixed number of restarts. The idea is to allow more restarts on instances for which the ILS has a fast convergence.

The main loop (lines 4–29) launches successive ILS and records the global best solution S^*. The first pair (S, T) of each ILS is computed in lines 5–8, starting with the heuristic *GiantTour* explained in the sequel. The ILS loop (lines 10–25) implements the cyclic alternation between the two search spaces, as explained previously, using the procedures *Perturbation*, *Split*, *LocalSearch*, and *Concatenate*. The counter *cont_f* is incremented when the current best solution of the ILS (S) is not improved, otherwise it is reset to zero. The ILS stops after *nb_f* iterations without improving its best solution S, after a total of *nb_ils* iterations, or when the budget is exhausted.

The relaxed version (R-ILS) is identical, except that lines 18 and 19 are moved before line 17 to always accept the new solution. This nonstandard ILS generates a trajectory in which the cost may temporarily increase, rather than staying on the incumbent solution S and applying perturbation and local search until a better solution is found.

6.2. Initial giant tours

The initial giant tour of each ILS is built by a method inspired by the geometric sweep heuristic for the CVRP (Laporte et al., 2000). The use of spatial (geographical) information is a tactical decision, particularly in the context of disaster logistics management. The heuristic initially computes the centroid of each cluster as follows: if (x_i, y_i) denotes the geographical coordinates of each node $i \in V\backslash\{0\}$, the centroid coordinates $(\bar{x}_j, \bar{y}_j)$ of each cluster $j = 1 \ldots m$ are given in Equation (5). Then, the polar angle of each centroid is computed, taking the depot as origin, and the clusters are sorted in increasing order of these angles to give an ordered list L. Finally, the giant tour starts at the depot, visits the ordered clusters, and returns to the depot.

$$\forall j = 1 \ldots m: \quad \bar{x}_j = \frac{\sum_{i \in C_j} x_i}{|C_j|} \quad \text{and} \quad \bar{y}_j = \frac{\sum_{i \in C_j} y_i}{|C_j|}. \tag{5}$$

To get a distinct tour for each restart, the first cluster visited is shifted and the list L is browsed circularly each time the *GiantTour* heuristic is called: the first giant tour is $(L_1, L_2, \ldots, L_n)$, the second one $(L_2, L_3, \ldots, L_n, L_1)$, and so on.

Algorithm 5 : MS-ILS pseudo-code

```
Input: nb_r, nb_ils, nb_f
cost(S*) ← ∞;
budget ← nb_r × nb_ils;
cont_b ← 0;
repeat
    GiantTour (T);
    S ← Split (T);
    LocalSearch (S);
    T ← Concatenate (S);
    cont_ils, cont_f ← 0;
    while ( (cont_ils < nb_ils) and (cont_f < nb_f) and (cont_b < budget) ) do
        cont_ils ← cont_ils + 1;
        T' ← T;
        Perturbation (T');
        S' ← Split (T');
        LocalSearch (S');
        cont_b ← cont_b + 1;
        if ( cost(S') < cost(S) ) then
            S ← S';
            T ← Concatenate (S);
            cont_f ← 0;
        end
        else
            cont_f ← cont_f + 1;
        end
    end
    if ( cost(S) < cost(S*) ) then
        S* ← S;
    end
until ( cont_b < budget );
```

6.3. Perturbation and local search

The perturbation is applied to giant tours, not to GVRP solutions. It consists in exchanging two randomly selected clusters.

Swap and relocation moves are used in the local search procedure that is applied to each new GVRP solution. A swap is an exchange of two clusters, while a relocation removes one cluster to reinsert it in a different position. The two moves are applied to one or two routes. Vehicle capacity must be checked for inter-route moves only. Swap moves are evaluated before relocations. The cost variation of each move is quickly computed, assuming that the active nodes of involved clusters

are preserved. As soon as an improving move is detected, it is accepted and the current solution is updated (first-improvement strategy). A further improvement can be obtained by recomputing for each affected route the best active nodes in visited clusters, using the dynamic programming method developed by Bontoux et al. (2010) for the GTSP and recalled in Section 5.1. This process is repeated until no improvement is possible.

Concerning complexity, the number of swap moves is $O(m^2)$ since they are applied to each pair of effective clusters (the cluster containing the depot is not considered). There are also $O(m^2)$ relocations, because the m effective clusters can be inserted in $O(m)$ different positions. Each move can be evaluated in $O(1)$. For a route containing k effective clusters $E_1, E_2, \ldots, E_k$, the complexity of the dynamic programming procedure applied after each accepted move is linear in the number of arcs $|E_1| + \sum_{i=1}^{k-1} |E_i| \cdot |E_{i+1}| + |E_k|$.

7. Computational experiments

The computational experiments were carried out on an Intel Core 2 Duo with 3 GHz clock and 4 GB of RAM. Five sets of benchmark instances were used. They are detailed in Section 7.1. The CG is coded in C++ and coupled with the MIP solver CPLEX 12.0 using default parameters. Its running time is limited to 1 hour. The MS-ILS and R-ILS metaheuristics have been developed in ANSI C. After preliminary experiments, both MS-ILS and R-ILS have been tuned with the same parameters: number of restarts $nb\_r = 20$, maximum number of iterations per ILS $nb\_ils = 100$, maximum number of iterations without improvement $nb\_f = 20$. The computing budget is then $budget = nb\_r \times nb\_ils = 2000$ calls to the local search.

7.1. Sets of instances

Among the five sets of benchmarks used in our tests, three consist of GVRP instances from published papers, one is derived from VRPTW instances (VRP with time windows), while the last one is a new set that uses a different cluster construction detailed below.

Pop et al. (2011) used instances *eil51*, *eil76*, and *eil101* from the TSPLIB, added demands and defined $\lceil n/5 \rceil$ clusters using a clustering approach proposed by Fischetti et al. (1997). However, as they handled the split GVRP, they selected small vehicle capacities to oblige several routes to visit the same cluster. They got seven instances using two capacities for *eil51*, three for *eil76*, and two for *eil101*. We kept the networks, demands, and clusters but set vehicle capacity to the maximum cluster demand, multiplied by $\alpha \in \{1.5, 2.0\}$. This gives the first set (P) with six instances, from 51 nodes and 11 clusters to 101 nodes and 21 clusters. The instance names contain the value of α, followed by the TSPLIB file name.

The second set (A) contains only one instance with 51 nodes and 24 clusters, derived by Araque et al. (1994) from a CVRP instance and used by Ghiani and Improta (2000).

The third set (S) reuses the networks and demands of three VRPTW instances with 101 nodes (*c101*, *r101*, and *rc101*) proposed by Solomon (1987). Time windows are dropped while

clusters and vehicle capacities are determined as for set (P). The six resulting instances contain 21 clusters.

The fourth set (B), adapted from the CVRP library available at http://branchandcut/VRP/data/, was used by Bektas et al. (2011) and Moccia et al. (2011). Bektas et al. (2011) solved these GVRP instances but with a fixed fleet. In this set composed of two subsets ($B1$) and ($B2$) of 74 instances each, instances have clusters with up to four sites. In the naming format X-nY-kZ-$C\Omega$-$V\Phi$, X corresponds to the instance type, Y to the number of vertices, Z to the number of vehicles in the original CVRP instance, Ω to the number of clusters, and Φ to the number of vehicles in the resulting GVRP instance. Set $B1$ contains instances with 32–101 nodes and 16–51 clusters. In set $B2$, the number of nodes varies in the same interval but there are 11–34 clusters.

The purpose of the comparison with the work of Bektas et al. (2011) is rather to convince decision makers that ignoring the constraint on fleet size does not lead to major changes in terms of cost and number of vehicles. As we will see, we can sometimes reduce the total duration or length of the routes at the expense of one additional vehicle, which is important in humanitarian logistics, and in particular in this research that takes place in a project on disaster response.

The fifth set (APS; for Afsar, Prins, and Santos) contains elongated clusters inspired by the valleys or peninsulas that can be found in disaster logistics, as mentioned in the introduction. It consists of 20 new instances randomly generated using a different cluster construction method. The region is divided into elementary squares, and some pairs of adjacent squares (vertically or horizontally) are randomly merged to get a mix of small squares and rectangles. The set is made of five groups of four instances, with a number of clusters $m \in \{12, 25, 50, 75, 100\}$. The four instances in each group comprise two subgroups of two instances: one with small clusters (at most three nodes per cluster on average) and one with larger clusters (at least five customers on average). Each subgroup of two includes one ST problem with short trips (two or three clusters per trip on average) and one LT with longer trips (five to seven clusters per trip). For many vehicle routing problems such as the CVRP, it is well known that instances with long trips are harder to solve, at least for exact methods. The resulting instances range from 12 clusters and 24 customers to 100 clusters and 504 customers, with a file name format m-n-ST or LT. The smallest one, 12-24-ST, is illustrated in Fig. 1.

Note that comparisons with published methods are not possible for sets (P) and (S) (instances are modified for our GVRP version) and for the set of new instances (APS). Solution costs can be compared for the unique instance of set (A) and, when our algorithms find the same number of vehicles as Bektas et al. (2011), for set (B).

7.2. Results

Tables 1–5 provide the numerical results. The instance names are followed by the results obtained using CG, MS-ILS, and R-ILS. Concerning CG, the best lower bound (LB), the best upper bound (UB) and the running time in seconds are provided. For each metaheuristic, the best solution value (UB), the running time in seconds, and the relative gap in percent to the lower bound produced by CG are listed. Asterisks denote proven optima. Null times represent durations smaller than 0.01

Table 1
Numerical results for Pop, Araque, Solomon, and *APS* instances

			Column generation			MS-ILS			R-ILS		
Instances	m	Q	*LB*	*UB*	$t(s)$	*UB*	$t(s)$	Gap(%)	*UB*	$t(s)$	Gap (%)
1.5-eil51	11	186	258.24	258.24*	0.24	258.24*	0.14	0.00	258.24*	0.13	0.00
1.5-eilA76	16	267	376.82	376.82*	0.15	379.17	0.30	0.62	376.82*	0.28	0.00
1.5-eilA101	21	320	414.86	414.86*	6.78	429.53	0.54	3.54	414.86*	0.52	0.00
2-eil51	11	248	239.53	239.53*	0.06	239.53*	0.17	0.00	239.53*	0.16	0.00
2-eilA76	16	356	329.31	329.31*	2.58	332.37	0.35	0.93	329.31*	0.35	0.00
2-eilA101	21	426	375.82	386.71	131.95	390.63	0.65	3.94	386.82	0.63	2.93
v51c24	24	15	541.00	541.00*	1772.58	541.00*	0.89	0.00	541.00*	0.73	0.00
1.5-c101	21	255	2476.90	2476.90*	0.29	2476.90*	0.51	0.00	2476.90*	0.47	0.00
2-c101	21	340	2383.69	2383.69*	0.97	2383.69*	0.56	0.00	2383.69*	0.51	0.00
1.5-rc101	21	300	761.02	761.02*	237.79	762.43	0.77	0.19	761.02*	0.51	0.00
2-rc101	21	400	682.89	682.89*	718.69	682.89*	0.65	0.00	682.89*	0.60	0.00
1.5-r101	21	320	624.86	624.86*	8.32	639.53	0.58	2.35	626.56	0.55	0.27
2-r101	21	426	596.68	596.68*	2137.32	600.63	0.63	0.66	596.82	0.63	0.02
12-24-ST	12	200	504.37	504.37*	0.01	504.37*	0.13	0.00	504.37*	0.11	0.00
12-24-LT	12	200	354.64	354.64*	5.28	354.64*	0.21	0.00	354.64*	0.19	0.00
12-72-ST	12	200	506.08	506.08*	0.17	506.08*	0.13	0.00	506.08*	0.13	0.00
12-72-LT	12	200	290.29	290.29*	28.26	290.29*	0.31	0.00	290.29*	0.30	0.00
25-54-ST	25	200	1006.29	1006.29*	19.49	1006.29*	0.79	0.00	1006.29*	0.48	0.00
25-54-LT	25	200	585.88	589.27	1969.87	589.27	0.80	0.58	589.27	0.68	0.58
25-162-ST	25	200	858.75	858.75*	327.76	860.57	0.91	0.21	858.75*	0.60	0.00
25-162-LT	25	200	560.88	593.16	1821.87	581.31	1.22	3.64	581.28	1.04	3.64
50-121-ST	50	500	1566.95	1580.72	77.60	1609.83	3.31	2.74	1595.50	2.12	1.82
50-121-LT	50	500	–	–	–	1150.81	4.22	–	1147.26	2.49	–
50-283-ST	50	500	1655.23	1655.23*	84.76	1682.58	3.64	1.65	1668.27	2.16	0.79
50-283-LT	50	500	–	–	–	1135.82	4.27	–	1120.69	2.86	–
75-150-ST	75	500	2788.43	2805.14	857.91	2830.27	14.16	1.50	2821.24	5.06	1.18
75-150-LT	75	500	–	–	–	1482.35	13.64	–	1461.64	6.81	–
75-400-ST	75	500	2643.42	2664.47	787.05	2693.37	13.34	1.89	2666.54	4.71	0.87
75-400-LT	75	500	–	–	–	1425.73	10.81	–	1398.16	6.24	–
100-216-ST	100	500	4130.81	4159.25	1230.29	4265.72	33.35	3.27	4213.20	9.32	1.99
100-216-LT	100	500	–	–	–	1972.63	29.45	–	1958.01	12.04	–
100-504-ST	100	500	3661.55	3676.79	1103.67	3763.73	22.52	2.79	3711.60	8.46	1.37
100-504-LT	100	500	–	–	–	2210.71	23.25	–	2182.75	10.64	–

second. Empty cells means that the linear program at the root of the B&P tree cannot be solved in 1 hour.

Table 1 depicts results for Pop, Araque, Solomon, and *APS* instances (33 instances). The CG method solves to optimality 19 instances in less than 36 minutes: All instances are of sets (P), (A), and (S), except instance 2-eilA101 that displays an optimality gap of 2.9%. Solutions with a maximum gap of 0.9% (for 50-121-ST) are obtained for all *APS* instances with ST. The biggest ST instance solved exactly is 50-283-ST (50 clusters and 283 nodes). As

expected, LT instances with longer routes look harder: no solution is found from 50 clusters onward.

For these test sets, MS-ILS and R-ILS, respectively, reach 11 and 16 optima of the 19 found by CG. For the others, when the CG lower bound is available, solutions are obtained in at most 34 seconds and the gaps never exceed 4%. R-ILS is always faster than MS-ILS and it provides strictly better solutions when MS-ILS is not optimal. It is important to mention that for some instances, such as 25-162-LT, the CG approach is not optimal, so the gaps computed using the CG lower bound overestimate the real gap.

As already mentioned, we have run our algorithms on the fourth test set (B) with limited fleet sizes to see if solutions with smaller mileage or duration, important in disaster logistics, could be obtained. Due to page length, results for subset ($B1$) are split over Tables 2 and 3, the ones for subset ($B2$) being listed in Tables 4 and 5. All these tables include additional columns for the results of Bektas et al. (2011): the fleet size K (minimum number of vehicles, computed using a bin-packing algorithm), and the upper bound UB found using an LNS. Columns K for MS-ILS and R-ILS indicate the number of vehicles used, while all other columns have been previously defined.

For our algorithms, an asterisk means that our version of the GVRP with a flexible fleet size is solved to optimality. Values in boldface indicate an improvement in terms of cost compared with results produced in Bektas et al. (2011), but sometimes at the expense of one more vehicle. Finally, costs in italics mean they are slightly worse than those found in Bektas et al. (2011).

On subset ($B1$), the CG method fails on 22 instances of 74, all with more than 50 nodes, but provides good results for the 52 others, including 12 optima. On these 52 instances, the same solutions as LNS are found 32 times (same cost, same number of vehicles), better costs are obtained 4 times but with one vehicle more, and a bit higher costs 16 times. MS-ILS finds 53 times the same solution as LNS, 4 times a better solution at the expense of an extra vehicle, and 17 times a slightly degraded cost but with the same vehicles (except on P-n60-k10-C30-V5 that requires one supplementary vehicle). For R-ILS, these numbers are, respectively, 65, 4, and 5. R-ILS always outperforms MS-ILS, except for the last instance of Table 3, while being substantially faster. Note that our two metaheuristics reach the 12 optima of CG.

For subset ($B2$), CG looks more robust. It fails only 10 times of 74 and gets 29 optima among the 64 other solutions. CG retrieves 42 LNS solutions, improves four costs (two with one additional vehicle and two without), and provides slightly degraded solutions in 18 cases. MS-ILS finds the same solutions as LNS for 60 instances. It provides better costs with the same vehicles in three cases, better costs with an extra vehicle in two cases, and augmented costs with the same number of vehicles in nine cases. These numbers become, respectively, 68, 3, 2, and 1 for R-ILS. All optima of CG are achieved by R-ILS while only one is missed by MS-ILS.

Concerning running times, the maximum duration of the CG approach is 36 minutes. The LNS lasts at most 1.87 seconds. Our two metaheuristics are also quite fast: they need more than 2.5 seconds only for the last instance of Table 3.

Summarizing, our approach and the algorithms of Bektas et al. (2011) look complementary. Our policy concerning the number of vehicles pays more attention to the application

Table 2
Numerical results for subset ($B1$) of Bektas instances

	Bektas		Column generation			MS-ILS				R-ILS			
Instances	K	UB	LB	UB	$t(s)$	UB	K	$t(s)$	Gap (%)	UB	K	$t(s)$	Gap (%)
A-n32-k5-C16-V2	2	519	489.67	**508**	24.07	**508**	3	0.34	3.74	**508**	3	0.31	3.74
A-n33-k5-C17-V3	3	451	441.00	451	12.03	451	3	0.42	2.27	451	3	0.36	2.27
A-n33-k6-C17-V3	3	465	462.50	465	3.61	465	3	0.37	0.54	465	3	0.31	0.54
A-n34-k5-C17-V3	3	489	485.83	489	18.87	489	3	0.42	0.65	489	3	0.37	0.65
A-n36-k5-C18-V2	2	505	492.18	**502**	496.57	**502**	3	0.46	2.00	**502**	3	0.35	2.00
A-n37-k5-C19-V3	3	432	432.00	432*	7.70	432*	3	0.55	0.00	432*	3	0.45	0.00
A-n37-k6-C19-V3	3	584	578.57	584	170.39	584	3	0.46	0.94	584	3	0.37	0.94
A-n38-k5-C19-V3	3	476	473.33	476	33.58	476	3	0.55	0.56	476	3	0.43	0.56
A-n39-k5-C20-V3	3	557	527.24	*559*	485.14	557	3	0.65	5.64	557	3	0.52	5.64
A-n39-k6-C20-V3	3	544	538.58	544	741.56	544	3	0.5	1.01	544	3	0.46	1.01
A-n44-k6-C22-V3	3	608	607.52	608	1957.23	608	3	0.64	0.08	608	3	0.45	0.08
A-n45-k6-C23-V4	4	613	587.25	613	220.37	613	4	0.74	4.38	613	4	0.66	4.38
A-n45-k7-C23-V4	4	674	640.00	*682*	825.87	674	4	0.79	5.31	674	4	0.58	5.31
A-n46-k7-C23-V4	4	593	571.25	593	1154.88	593	4	0.71	3.81	593	4	0.56	3.81
A-n48-k7-C24-V4	4	667	644.00	*669*	1174.62	*668*	4	0.81	3.73	667	4	0.66	3.57
A-n53-k7-C27-V4	4	603	597.70	603	215.55	603	4	1.20	0.89	603	4	0.87	0.89
A-n54-k7-C27-V4	4	690	669.48	*693*	150.03	690	4	1.02	3.07	690	4	0.77	3.07
A-n55-k9-C28-V5	5	699	675.12	*701*	42.94	699	5	1.13	3.54	699	5	0.85	3.54
A-n60-k9-C30-V5	5	769	–	–	–	769	5	1.44	–	769	5	0.87	–
A-n61-k9-C31-V5	5	638	624.67	638	2132.60	*639*	5	1.24	2.29	638	5	0.99	2.13
A-n62-k8-C31-V4	4	740	–	–	–	740	4	1.61	–	740	4	1.09	–
A-n63-k9-C32-V5	5	912	–	–	–	912	5	1.52	–	912	5	1.04	–
A-n63-k10-C32-V5	5	801	–	–	–	801	5	1.33	–	801	5	1.01	–
A-n64-k9-C32-V5	5	763	–	–	–	*764*	5	1.50	–	763	5	1.04	–
A-n65-k9-C33-V5	5	682	–	–	–	682	5	1.53	–	682	5	1.29	–
A-n69-k9-C35-V5	5	680	–	–	–	*689*	5	2.22	–	680	5	1.53	–
A-n80-k10-C40-V5	5	997	–	–	–	997	5	2.40	–	997	5	1.75	–
B-n31-k5-C16-V3	3	441	377.80	*447*	449.20	441	3	0.45	16.73	441	3	0.34	16.73
B-n34-k5-C17-V3	3	472	435.85	*477*	380.79	472	3	0.49	8.29	472	3	0.42	8.29
B-n35-k5-C18-V3	3	626	584.46	*647*	2007.23	626	3	0.44	7.11	626	3	0.45	7.11
B-n38-k6-C19-V3	3	451	444.67	451	96.38	451	3	0.56	1.42	451	3	0.41	1.42
B-n39-k5-C20-V3	3	357	350.00	357	40.24	357	3	0.72	2.00	357	3	0.62	2.00
B-n41-k6-C21-V3	3	481	475.15	481	39.82	481	3	0.70	1.23	481	3	0.62	1.23
B-n43-k6-C22-V3	3	483	458.25	*499*	1237.85	483	3	0.65	5.40	483	3	0.58	5.40
B-n44-k7-C22-V4	4	540	498.49	*558*	1294.98	540	4	0.63	8.33	540	4	0.57	8.33
B-n45-k5-C23-V3	3	497	493.39	*498*	124.81	497	3	0.81	0.73	497	3	0.79	0.73
B-n45-k6-C23-V4	4	478	445.49	478	1066.63	478	4	0.95	7.30	478	4	0.70	7.30
B-n50-k7-C25-V4	4	449	449.00	449*	70.51	449*	4	1.16	0.00	449*	4	0.96	0.00
B-n50-k8-C25-V5	5	916	–	–	–	*922*	5	0.87	–	*917*	5	0.62	–
B-n51-k7-C26-V4	4	651	634.60	*654*	1081.40	651	4	0.97	2.58	651	4	0.82	2.58
B-n52-k7-C26-V4	4	450	–	–	–	450	4	1.04	–	450	4	0.90	–
B-n56-k7-C28-V4	4	486	–	–	–	486	4	1.29	–	486	4	1.05	–
B-n57-k7-C29-V4	4	751	–	–	–	751	4	1.54	–	751	4	1.09	–
B-n57-k9-C29-V5	5	942	–	–	–	*947*	5	1.15	–	*944*	5	0.88	–
B-n63-k10-C32-V5	5	816	–	–	–	816	5	1.53	–	816	5	1.12	–

Continued

Table 2
Continued

	Bektas		Column generation			MS-ILS				R-ILS			
Instances	*K*	*UB*	*LB*	*UB*	*t(s)*	*UB*	*K*	*t(s)*	Gap (%)	*UB*	*K*	*t(s)*	Gap (%)
B-n64-k9-C32-V5	5	509	–	–	–	509	5	1.61	–	509	5	1.42	–
B-n66-k9-C33-V5	5	808	–	–	–	*809*	5	1.66	–	*809*	5	1.28	–
B-n67-k10-C34-V5	5	673	–	–	–	673	5	1.72	–	673	5	1.40	–
B-n68-k9-C34-V5	5	704	–	–	–	*705*	5	1.71	–	704	5	1.40	–
B-n78-k10-C39-V5	5	803	–	–	–	*809*	5	2.43	–	803	5	1.79	–
P-n16-k8-C8-V5	5	239	239.00	239*	0.00	239*	5	0.04	0.00	239*	5	0.04	0.00
P-n19-k2-C10-V2	2	147	147.00	147*	8.09	147*	2	0.15	0.00	147*	2	0.13	0.00
P-n20-k2-C10-V2	2	154	154.00	154*	15.90	154*	2	0.16	0.00	154*	2	0.14	0.00
P-n21-k2-C11-V2	2	160	160.00	160*	57.42	160*	2	0.19	0.00	160*	2	0.16	0.00
P-n22-k2-C11-V2	2	162	162.00	162*	32.81	162*	2	0.18	0.00	162*	2	0.16	0.00
P-n22-k8-C11-V5	5	314	314.00	314*	0.06	314*	5	0.12	0.00	314*	5	0.12	0.00
P-n23-k8-C12-V5	5	312	312.00	312*	9.00	312*	5	0.13	0.00	312*	5	0.11	0.00
P-n40-k5-C20-V3	3	294	294.00	294*	478.65	294*	3	0.59	0.00	294*	3	0.53	0.00
P-n45-k5-C23-V3	3	337	337.00	337*	1002.55	337*	3	0.82	0.00	337*	3	0.68	0.00
P-n50-k10-C25-V5	5	410	404.00	*412*	4.9	*413*	5	0.68	2.23	410	5	0.60	1.49
P-n50-k7-C25-V4	4	353	341.30	353	35.00	353	4	0.80	3.43	353	4	0.77	3.43
P-n50-k8-C25-V4	4	392	370.00	**372**	9.10	**373**	5	0.81	0.81	**372**	5	0.67	0.54
P-n51-k10-C26-V6	6	427	427.00	427*	0.89	427*	6	0.82	0.00	427*	6	0.58	0.00
P-n55-k10-C28-V5	5	415	407.22	415	10.38	415	5	0.94	1.91	415	5	0.81	1.91

Table 3
Numerical results for subset (*B*1) of Bektas instances (continued)

	Bektas		Column generation			MS-ILS				R-ILS			
Instances	*K*	*UB*	*LB*	*UB*	*t(s)*	*UB*	*K*	*t(s)*	Gap (%)	*UB*	*K*	*t(s)*	Gap (%)
P-n55-k15-C28-V8	8	555	546.32	**551**	2.05	**551**	9	0.81	0.86	**551**	9	0.61	0.86
P-n55-k7-C28-V4	4	361	346.89	361	225.13	361	4	1.13	4.07	361	4	0.95	4.07
P-n55-k8-C28-V4	4	361	351.86	*363*	105.94	361	4	1.06	2.60	361	4	0.91	2.60
P-n60-k10-C30-V5	5	443	431.75	*447*	19.98	*445*	6	1.18	3.07	*445*	5	0.97	3.07
P-n60-k15-C30-V8	8	565	557.17	565	3.02	*566*	8	1.06	1.58	565	8	0.80	1.41
P-n65-k10-C33-V5	5	487	485.11	*489*	1280.91	*490*	5	1.38	1.01	487	5	1.04	0.39
P-n70-k10-C35-V5	5	485	484.91	485	1542.81	*486*	5	1.57	0.22	485	5	1.24	0.02
P-n76-k4-C38-V2	2	383	–	–	–	*390*	2	2.35	–	383	2	1.92	–
P-n76-k5-C38-V3	3	405	–	–	–	*413*	3	2.33	–	405	3	1.96	–
P-n101-k4-C51-V2	2	455	–	–	–	*456*	2	6.19	–	*457*	2	3.95	–

context—disaster logistics. We prefer to avoid fixing the number of vehicles *a priori* to focus on solution cost (mileage or duration). This approach looks profitable, since in most cases the authorities will need a minimal fleet and obtain sometimes better costs by investing in one additional vehicle.

Table 4
Numerical results for subset (*B*2) of Bektas instances

	Bektas		Column generation			MS-ILS				R-ILS			
Instances	*K*	*UB*	*LB*	*UB*	*t(s)*	*UB*	*K*	*t(s)*	Gap*(%)*	*UB*	*K*	*t(s)*	Gap *(%)*
A-n32-k5-C11-V2	2	386	386.00	386*	13.64	386*	2	0.20	0.00	386*	2	0.17	0.00
A-n33-k5-C11-V2	2	318	313.50	**316**	0.69	**315**	2	0.20	0.48	**315**	2	0.18	0.48
A-n33-k6-C11-V2	2	370	370.00	370*	1.28	370*	2	0.18	0.00	370*	2	0.16	0.00
A-n34-k5-C12-V2	2	419	408.80	*422*	1.72	419	2	0.21	2.50	419	2	0.20	2.50
A-n36-k5-C12-V2	2	396	396.00	396*	118.09	396*	2	0.22	0.00	396*	2	0.20	0.00
A-n37-k5-C13-V2	2	347	347.00	347*	0.49	347*	2	0.30	0.00	347*	2	0.27	0.00
A-n37-k6-C13-V2	2	431	431.00	431*	40.18	431*	2	0.23	0.00	431*	2	0.19	0.00
A-n38-k5-C13-V2	2	367	367.00	367*	1.62	367*	2	0.28	0.00	367*	2	0.28	0.00
A-n39-k5-C13-V2	2	364	364.00	364*	82.30	364*	2	0.27	0.00	364*	2	0.25	0.00
A-n39-k6-C13-V2	2	403	403.00	403*	2.18	403*	2	0.24	0.00	403*	2	0.21	0.00
A-n44-k6-C15-V2	2	503	491.00	**491***	24.46	**491***	3	0.34	0.00	**491***	3	0.33	0.00
A-n45-k6-C15-V3	3	474	474.00	474*	43.03	474*	3	0.34	0.00	474*	3	0.31	0.00
A-n45-k7-C15-V3	3	475	460.20	475	39.55	475	3	0.35	3.22	475	3	0.29	3.22
A-n46-k7-C16-V3	3	462	451.00	*464*	13.48	462	3	0.36	2.44	462	3	0.34	2.44
A-n48-k7-C16-V3	3	451	442.00	451	68.12	451	3	0.40	2.04	451	3	0.34	2.04
A-n53-k7-C18-V3	3	440	433.50	440	70.92	440	3	0.55	1.50	440	3	0.47	1.50
A-n54-k7-C18-V3	3	482	479.25	*483*	809.83	482	3	0.52	0.57	482	3	0.42	0.57
A-n55-k9-C19-V3	3	473	473.00	473*	8.24	473*	3	0.53	0.00	473*	3	0.40	0.00
A-n60-k9-C20-V3	3	595	583.50	*596*	1190.53	595	3	0.57	1.97	595	3	0.47	1.97
A-n61-k9-C21-V4	4	473	469.36	*478*	37.78	473	4	0.59	0.78	473	4	0.56	0.78
A-n62-k8-C21-V3	3	596	579.54	*616*	1540.94	596	3	0.68	2.84	596	3	0.55	2.84
A-n63-k10-C21-V4	4	593	588.20	*599*	191.38	593	4	0.68	0.82	593	4	0.53	0.82
A-n63-k9-C21-V3	3	642	629.84	642	647.42	*644*	3	0.67	2.25	642	3	0.48	1.93
A-n64-k9-C22-V3	3	536	529.36	*546*	1464.32	536	3	0.75	1.25	536	3	0.66	1.25
A-n65-k9-C22-V3	3	500	500.00	500*	69.23	500*	3	0.62	0.00	500*	3	0.6	0.00
A-n69-k9-C23-V3	3	520	514.20	*521*	312.42	520	3	0.77	1.13	520	3	0.71	1.13
A-n80-k10-C27-V4	4	710	–	–	–	710	4	1.07	–	710	4	0.85	–
B-n31-k5-C11-V2	2	356	314.00	*357*	10.18	356	2	0.22	13.38	356	2	0.17	13.38
B-n34-k5-C12-V2	2	369	342.33	369	6.84	369	2	0.24	7.79	369	2	0.23	7.79
B-n35-k5-C12-V2	2	501	455.80	*510*	10.92	501	2	0.24	9.92	501	2	0.25	9.92
B-n38-k6-C13-V2	2	370	370.00	370*	1.35	370*	2	0.27	0.00	370*	2	0.26	0.00
B-n39-k5-C13-V2	2	280	280.00	280*	0.57	280*	2	0.32	0.00	280*	2	0.29	0.00
B-n41-k6-C14-V2	2	407	387.75	*412*	9.69	407	2	0.32	4.96	407	2	0.33	4.96
B-n43-k6-C15-V2	2	343	342.80	*344*	24.78	343	2	0.36	0.06	343	2	0.30	0.06
B-n44-k7-C15-V3	3	395	381.00	395	14.32	395	3	0.33	3.67	395	3	0.29	3.67
B-n45-k5-C15-V2	2	422	407.29	**412**	7.11	**410**	2	0.38	0.67	**410**	2	0.40	0.67
B-n45-k6-C15-V2	2	336	–	–	–	336	2	0.37	–	336	2	0.35	–
B-n50-k7-C17-V3	3	393	383.50	*397*	9.75	393	3	0.48	2.48	393	3	0.47	2.48
B-n50-k8-C17-V3	3	598	590.24	598	125.49	598	3	0.40	1.31	598	3	0.31	1.31
B-n51-k7-C17-V3	3	511	501.33	511	30.83	511	3	0.48	1.93	511	3	0.45	1.93
B-n52-k7-C18-V3	3	359	348.00	359	12.27	359	3	0.55	3.16	359	3	0.53	3.16
B-n56-k7-C19-V3	3	356	346.00	*357*	436.08	356	3	0.66	2.89	356	3	0.55	2.89
B-n57-k7-C19-V3	3	558	517.67	*576*	308.50	558	3	0.65	7.79	558	3	0.55	7.79
B-n57-k9-C19-V3	3	681	–	–	–	681	3	0.54	–	681	3	0.49	–
B-n63-k10-C21-V3	3	599	599.00	599*	1260.49	599*	3	0.70	0.00	599*	3	0.55	0.00
B-n64-k9-C22-V4	4	452	447.20	452	54.33	452	4	0.73	1.07	452	4	0.67	1.07

Table 5
Numerical results for subset (B2) of Bektas instances (continued)

	Bektas		Column generation			MS-ILS				R-ILS			
Instances	*K*	*UB*	*LB*	*UB*	*t(s)*	*UB*	*K*	*t(s)*	Gap (%)	*UB*	*K*	*t(s)*	Gap (%)
B-n66-k9-C22-V3	3	609	–	–	–	609	3	0.84	–	609	3	0.64	–
B-n67-k10-C23-V4	4	558	–	–	–	558	4	0.80	–	558	4	0.77	–
B-n68-k9-C23-V3	3	523	–	–	–	523	3	0.82	–	523	3	0.66	–
B-n78-k10-C26-V4	4	606	–	–	–	606	4	0.99	–	606	4	0.87	–
P-n16-k8-C6-V4	4	170	170.00	170*	0.17	170*	4	0.03	0.00	170*	4	0.03	0.00
P-n19-k2-C7-V1	1	111	111.00	111*	0.03	111*	1	0.09	0.00	111*	1	0.09	0.00
P-n20-k2-C7-V1	1	117	117.00	117*	0.05	117*	1	0.08	0.00	117*	1	0.08	0.00
P-n21-k2-C7-V1	1	117	117.00	117*	0.04	117*	1	0.08	0.00	117*	1	0.08	0.00
P-n22-k2-C8-V1	1	111	111.00	111*	0.04	111*	1	0.11	0.00	111*	1	0.10	0.00
P-n22-k8-C8-V4	4	249	249.00	249*	0.00	249*	4	0.00	0.00	249*	4	0.06	0.00
P-n23-k8-C8-V3	3	174	174.00	174*	0.01	174*	3	0.07	0.00	174*	3	0.07	0.00
P-n40-k5-C14-V2	2	213	213.00	213*	7.17	213*	2	0.30	0.00	213*	2	0.27	0.00
P-n45-k5-C15-V2	2	238	238.00	238*	5.28	238*	2	0.31	0.00	238*	2	0.34	0.00
P-n50-k7-C17-V3	3	261	261.00	261*	57.75	261*	3	0.41	0.00	261*	3	0.46	0.00
P-n50-k8-C17-V3	3	262	262.00	262*	0.57	262*	3	0.36	0.00	262*	3	0.41	0.00
P-n50-k10-C17-V4	4	292	292.00	292*	4.57	*293*	4	0.38	0.34	292*	4	0.36	0.00
P-n51-k10-C17-V4	4	309	309.00	309*	154.52	309*	4	0.35	0.00	309*	4	0.41	0.00
P-n55-k7-C19-V3	3	271	260.67	*272*	22.53	*272*	3	0.50	4.35	271	3	0.62	3.96
P-n55-k8-C19-V3	3	274	266.00	274	21.43	*275*	3	0.51	3.38	274	3	0.55	3.01
P-n55-k10-C19-V4	4	301	293.20	301	4.20	*302*	4	0.46	3.00	301	4	0.50	2.66
P-n55-k15-C19-V6	6	378	378.00	378*	0.13	378*	6	0.32	0.00	378*	6	0.32	0.00
P-n60-k10-C20-V4	4	325	315.50	*327*	5.12	*327*	4	0.49	3.65	325	4	0.53	3.01
P-n60-k15-C20-V5	5	382	368.67	**374**	1.67	**376**	6	0.41	1.99	**374**	6	0.39	1.45
P-n65-k10-C22-V4	4	372	361.80	372	14.49	*376*	4	0.59	3.92	372	4	0.62	2.82
P-n70-k10-C24-V4	4	385	375.80	385	31.27	*386*	4	0.73	2.71	385	4	0.71	2.45
P-n76-k4-C26-V2	2	320	–	–	–	**309**	2	1.11	–	**309**	2	1.04	–
P-n76-k5-C26-V2	2	309	–	–	–	309	2	1.03	–	309	2	1.04	–
P-n101-k4-C34-V2	2	374	–	–	–	*375*	2	2.32	–	*375*	2	1.91	–

8. Conclusions

In this paper, we proposed for the GVRP-flex, an exact solution method based on Dantzig–Wolfe decomposition and CG, and two metaheuristics derived from ILS. These algorithms can be useful for a problem that has still a scarce literature, in spite of promising applications in disaster relief operations.

The results show that the exact approach performs particularly well on compact clusters. The partial exploration of the B&P tree returns relatively small optimality gaps when demands are rather homogeneous and vehicle capacity is not very large.

This work also brings important contributions in terms of heuristics. A tour splitting procedure has been designed for the GVRP-flex and included in the metaheuristics. It allows searching the space of giant tours instead of the wider set of GVRP-flex solutions. The initial giant tours are built using a constructive heuristic based on geographical information.

The proposed metaheuristics produce high-quality results in a few seconds. R-ILS outperforms MS-ILS in terms of cost and running time, although it does not comply with the standard ILS model: it moves to a new local optimum even if it deteriorates the incumbent one. The better performance probably resides in the fact that accepted solutions contain optimal subsequences, since these are local optima relatively close to the incumbent solution (because of the small perturbation applied). This strategy avoids being trapped in a few attraction basins.

We plan to address other objective functions to better model emergency and equity in disaster interventions, and also integrate stochastic issues to tackle uncertainty. We also intend to couple the metaheuristic approach with the exact method to guide the search.

References

Altay, N., Green, W.G., III, 2006. OR/MS research in disaster operations management. *European Journal of Operational Research* 175, 475–493.

Araque, J.R., Kudva, G., Morin, T.L., Pekny, J.F., 1994. A branch-and-cut algorithm for vehicle routing problems. *Annals of Operations Research* 50, 37–59.

Baldacci, R., Bartolini, E., Laporte, G., 2010. Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society* 61, 1072–1077.

Barbarosoglu, G., Özdamar, L., Çevik, A., 2002. An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations. *European Journal of Operational Research* 140, 118–133.

Beasley, J.E., 1983. Route-first cluster-second methods for vehicle routing. *Omega* 11, 403–408.

Bektas, T., Erdogan, G., Ropke, S., 2011. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science* 45, 299–316.

Berkoune, D., Renaud, J., Rekik, M., Ruiz, A., 2012. Transportation in disaster response operations. *Socio-Economic Planning Sciences* 46, 23–32.

Bontoux, B., Artigues, C., Feillet, D., 2010. A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. *Computers & Operations Research* 37, 1844–1852.

Campbell, A.M., Vandenbussche, D., Hermann, W., 2008. Routing under relief efforts. *Transportation Science* 42, 127–145.

Caunhye, A.M., Nie, X., Pokharel, S., 2012. Optimization models in emergency logistics: a literature review. *Socio-Economic Planning Sciences* 46, 4–13.

Coffrin, C., Van Hentenryck, P., Bent, R., 2011. Strategic stockpiling of power system supplies. Proceedings of the 2011 IEEE Power & Energy Society General Meetings (PES 2011), Detroit, MI.

Cullen, F., Jarvis, J., Ratliff, D., 1981. Set partitioning based heuristics for interactive routing. *Networks* 11, 125–144.

de la Torre, L.E., Dolinskaya, I.S., Smilowitz, K.R., 2012. Disaster relief routing: Integrating research and practice. *Socio-Economic Planning Sciences* 46, 88–97.

Desrosiers, J., Soumis, F., Desrochers, M., 1984. Routing with time windows by column generation. *Networks* 14, 545–565.

Dror, M., Langevin, A., 2000. Transformations and exact node routing solutions by column generation. In Dror, M. (ed.) *Arc Routing: Theory, Solutions and Applications*, Kluwer, Dordrecht, pp. 277–326.

Feillet, D., Dejax, P., Gendreau, M., Guéguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks* 44, 216–229.

Fischetti, M., González, J.J.S., Toth, P., 1997. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research* 45, 378–394.

Ghiani, G., Improta, G., 2000. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research* 122, 11–17.

Huang, M., Smilowitz, K.R., Balcik, B., 2012. Models for relief routing: equity, efficiency and efficacy. *Transportation Research Part E* 48, 2–18.

Kara, I., Bektas, T., 2003. Integer linear programming formulation of the generalized vehicle routing problem. Proceedings of the 5th EURO/INFORMS Joint International Meeting (2003), Istanbul, Turkey.

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

Lin, Y.H., Batta, R., Rogerson, P.A., Blatt, A., Flanigan, M., 2011. A logistic model for emergency supply of critical items in the aftermath of a disaster. *Socio-Economic Planning Sciences* 45, 132–145.

Lourenço, H.R., Martin, O.C., Stutzle, T., 2002. Iterated local search. In Glover, F. and Kochenberger, G. (eds) *Handbook of Metaheuristics*, Kluwer Academic Publishers, Boston, MA, pp. 321–353.

Moccia, L., Cordeau, J.-F., Laporte, G., 2011. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society* 63, 2, 232–244.

Ngueveu, S.U., Prins, C., Calvo, R.W., 2010. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research* 37, 1877–1885.

Nguyen, V.-P., Prins, C., Prodhon, C., 2012. A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. *Engineering Applications of Artificial Intelligence* 25, 56–71.

Pop, P.C., Matei, O., Sitar, C.P., Chira, C., 2010. A genetic algorithm for solving the generalized vehicle routing problem. In Corchado, E., Romay, M.G. and Savio, A.M. (eds) *Hybrid Artificial Intelligence Systems*, Vol. 6077, Lecture Notes in Computer Science. Springer, Berlin and Heidelberg, pp. 119–126.

Pop, P.C., Sitar, C.P., Zelina, I., Lupse, V., Chira, C., 2011. Heuristic algorithms for solving the generalized vehicle routing problem. *International Journal of Computers Communications & Control* 6, 158–165.

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

Prins, C., 2009. A GRASP×evolutionary local search hybrid for the vehicle routing problem. In Pereira, F.B. and Tavares, J. (eds) *Bio-Inspired Algorithms for the Vehicle Routing Problem*, Vol. 161, Studies in Computational Intelligence, Springer, Berlin and Heidelberg, pp. 35–53.

Rath, S., Gutjhar, W.J., 2011. A math-heuristic for the warehouse location-routing problem in disaster relief. *Computers & Operations Research*. doi: 10.1016/j.cor.2011.07.016.

Ribeiro, G.M., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research* 39, 3, 728–735.

Saadatseresht, M., Mansourian, A., Taleai, M., 2009. Evacuation planning using multiobjective evolutionary optimization approach. *European Journal of Operational Research* 198, 305–314.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35, 254–265.

Van Hentenryck, P., Coffrin, C., Bent, R., 2011. Vehicle routing for the last mile of power system restoration. Proceedings of the 17th Power Systems Computation Conference (PSCC'11), Stockholm, Sweden.

Yi, W., Özdamar, L., 2007. A dynamic logistics coordination model for evacuation and support in disaster response activities. *European Journal of Operational Research* 3, 1177–1193.