J Heuristics (2014) 20:483–509
DOI 10.1007/s10732-014-9247-0

A hybrid Granular Tabu Search algorithm for the Multi-Depot Vehicle Routing Problem

John Willmer Escobar · Rodrigo Linfati · Paolo Toth · Maria G. Baldoquin

Received: 22 November 2012 / Revised: 12 April 2014 / Accepted: 18 April 2014 /
Published online: 10 May 2014

Abstract In this paper, we propose a hybrid Granular Tabu Search algorithm to solve the *Multi-Depot Vehicle Routing Problem* (MDVRP). We are given on input a set of identical vehicles (each having a capacity and a maximum duration), a set of depots, and a set of customers with deterministic demands and service times. The problem consists of determining the routes to be performed to fulfill the demand of the customers by satisfying, for each route, the associated capacity and maximum duration constraints. The objective is to minimize the sum of the traveling costs related to the performed routes. The proposed algorithm is based on a heuristic framework previously introduced by the authors for the solution of the *Capacitated Location Routing Problem* (CLRP). The algorithm applies a hybrid Granular Tabu Search procedure, which considers different neighborhoods and diversification strategies, to improve the initial solution obtained by a hybrid procedure. Computational experiments on benchmark instances from the literature show that the proposed algorithm is able to produce, within short computing time, several best solutions obtained by the previously published methods and new best solutions.

J.W. Escobar · P. Toth (✉)
DEI Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi", University of Bologna, Bologna, Italy
e-mail: paolo.toth@unibo.it

J.W. Escobar
e-mail: johnwillmer.escobar2@unibo.it; jwescobar@javerianacali.edu.co

J.W. Escobar · M.G. Baldoquin
Departamento de Ingeniería Civil e Industrial, Pontificia Universidad Javeriana, Cali, Colombia
e-mail: mgulnara@javerianacali.edu.co

R. Linfati
Departamento de Ingeniería Industrial, Universidad del Bío-Bío, Concepción, Chile
e-mail: rlinfati@ubiobio.cl

Keywords Multi-Depot Vehicle Routing Problem · Granular Tabu Search · Heuristic algorithms

1 Introduction

This paper presents a hybrid heuristic algorithm for the *Multi-Depot Vehicle Routing Problem* (MDVRP). The MDVRP can be defined as follows: let $G = (V, E)$ be an undirected complete graph, where V is the vertex set and E the edge set. The vertex set V is partitioned into a subset $I = \{1, ..., m\}$ of depots and a subset $J = \{1, ..., n\}$ of customers. Each customer $j \in J$ has a nonnegative demand d_j and a nonnegative service time δ_j. Each depot $i \in I$ has a service time $\delta_i = 0$. It is to note that in the MDVRP not all the depots are necessarily used. A set of k identical vehicles, each with capacity Q, is available at each depot i. Each edge $(i, j) \in E$ has an associated nonnegative traveling cost c_{ij}. The goal of the MDVRP is to determine the routes to be performed to fulfill the demand of all the customers with the minimum traveling cost. The MDVRP is subject to the following constraints:

- Each route must start and finish at the same depot;
- Each customer must be visited exactly once by a single route;
- The total demand of each route must not exceed the vehicle capacity Q;
- The number of routes associated with each depot must not exceed the value of k.
- The total duration of each route (given by the sum of the traveling costs of the traversed edges and of the service times of the visited customers) must not exceed a given value D.

The MDVRP is known to be NP-hard, since it is a generalization of the well known *Capacitated Vehicle Routing Problem* (VRP), arising when $m = 1$. Exact algorithms were proposed by Laporte et al. (1984) and, recently, by Baldacci and Mingozzi (2009) and by Contardo and Martinelli (2014). Laporte et al. (1988) proposed an exact algorithm for the asymmetric case of the MDVRP (arising when G is a directed graph). These exact approaches can consistently solve to proven optimality small-medium size instances. For this reason, heuristic and metaheuristic algorithms have been proposed to solve successfully large MDVRP instances.

Early heuristics for the MDVRP have been proposed by Wren and Holliday (1972), Gillett and Johnson (1976), Gillett and Miller (1974), Golden et al. (1977), and Raft (1982). All these methods use adaptations of VRP algorithms to solve the MDVRP. Chao et al. (1993) proposed a multi-phase heuristic which is able to find good results with respect to the previously published approaches. In this work, customers are assigned to their closest depot. Then, a VRP is solved for each depot by using a modified savings algorithm proposed by Golden et al. (1977). Finally, the current solution is improved by using a method based on a record-to-record approach proposed in Dueck (1993).

Renaud et al. (1996b) proposed a tabu search heuristic which is able to find good results within short computing times. The algorithm first constructs an initial solution by assigning each customer to its nearest depot and by solving the VRP corresponding to each depot by using an improved petal heuristic described in Renaud et al. (1996a).

Finally, the tabu search considers three phases: fast improvement, intensification, and diversification. Each of these phases uses several inter-route and intra-route moves.

Cordeau et al. (1997) proposed a general tabu search heuristic which is also able to solve the *Periodic Vehicle Routing Problem* (PVRP) and the *Periodic Traveling Salesman Problem* (PTSP). The initial solution is constructed by assigning each customer to its nearest depot and by applying a procedure based on the GENI heuristic (for further details see Gendreau et al. (1992)). Infeasible solutions are allowed during the tabu search. For each infeasible solution, a penalty term proportional to the total excess quantity and to the excess duration of the routes is added.

Pisinger and Ropke (2007) proposed a unified heuristic, which is able to solve five different variants of the Vehicle Routing Problem. The MDVRP is solved by using an adaptive large neighborhood search (ALNS) algorithm. The ALNS is based on the large neighborhood search approach proposed by Shaw (1998), and the Ruin and Recreate paradigm introduced by Schrimpf et al. (2000).

Evolutionary approaches for the MDVRP have been proposed by Thangiah and Salhi (2001), Ombuki-Berman and Hanshar (2009), and Vidal et al. (2012). Vidal et al. (2012) proposed a metaheuristic based on the exploitation of a new population-diversity management mechanism to allow a broader access to reproduction, while preserving the memory of good solutions represented by the elite individuals of a population, and of an efficient offspring education scheme that integrates key features from efficient neighborhood search procedures such as memories and Granular Tabu Search concepts.

A parallel iterated tabu search heuristic has been developed by Cordeau and Maischberger (2012). This heuristic combines tabu search with a simple perturbation procedure to allow the algorithm to explore new parts of the solution space.

Subramanian et al. (2013) proposed a hybrid algorithm for a class of vehicle routing problems with homogeneous fleet. This algorithm combines an exact procedure based on the set partitioning (SP) formulation with an iterated local search (ILS) based heuristic.

Recently, Gulczynski et al. (2013), and Vidal et al. (2014) considered a relaxed version of the MDVRP where no limit is imposed on the number of routes associated with each depot (i.e. with $k = \infty$). For this version, Gulczynski et al. (2013) used integer programming to improve the current solution by relocating strings of consecutive customers on routes. Vidal et al. (2014) introduced a dynamic programming methodology for efficiently evaluating compound neighborhoods by considering the optimal assignment of vehicles and depots, and the optimal choice of the first customer to be visited on a performed route. An iterated local search procedure and a hybrid genetic algorithm are applied by considering these concepts.

In this paper, we propose a hybrid Granular Tabu Search algorithm, called ELTG, to solve the MDVRP. Algorithm ELTG is based on the heuristic framework introduced by Escobar et al. (2013) for the *Capacitated Location Routing Problem* (CLRP). The CLRP is a variant of the MDVRP in which several depots are available (each with a *capacity* and an *opening cost*), and the problem consists of determining the depots to be opened, and the routes to be performed for each open depot, so as to minimise the global cost, given by the sum of the costs of the open depots and the traveling costs of the routes, by satisfying the capacity constraints on the global demand of each route and of each open depot. In the CLRP, no constraint on the maximum duration of each route is imposed. Algorithm ELTG extends to the MDVRP the procedures proposed in Escobar et al. (2013) for the CLRP, so as to take into account the different characteristics of the two problems.

The paper is organized as follows. In Sect. 2 we give the detailed description of algorithm ELTG. Although some procedures are similar to the corresponding ones presented in Escobar et al. (2013), for the sake of clarity we prefer to give all the details of the procedures used in algorithm ELTG. Experimental results on the benchmark instances from the literature are reported in Sect. 3. Finally, conclusions and future research are presented in Sect. 4.

2 Hybrid Granular Tabu Search algorithm

The proposed algorithm is based on the Granular Tabu Search (GTS) idea for the VRP introduced by Toth and Vigo (2003). The GTS approach uses restricted neighborhoods, called *granular neighborhoods*, obtained from a *sparse graph* which includes all the edges with a cost not greater than a *granularity threshold value* $\vartheta = \beta \bar{z}$ (where β is a *sparsification factor* and $\bar{z}$ is the average cost of the edges), the edges belonging to the best feasible solution, and the edges (i, j) incident to the depots for which the *distance factor* $\varphi_{ij} = 2c_{ij} + \delta_j (\forall i \in I, j \in J)$ is not greater than the maximum duration D. The main objective of the GTS approach is to obtain high quality solutions within short computing times.

Algorithm ELTG applies three diversification strategies implemented to allow the exploration of new parts of the solution space. The first diversification strategy is based on the granularity diversification proposed in Toth and Vigo (2003). The second strategy is based on a penalty approach proposed by Gendreau et al. (1994) and Taillard (1993). The third diversification strategy determines every $N_{div} \times n$ iterations (where N_{div} is a given parameter) a feasible solution by using, for each depot, a local search procedure, called VRPH, which applies iteratively the VRP routines *vrp_sa, vrp_rtr* and *vrp_ej* proposed in Groer et al. (2010), until no improvement is reached. Procedure VRPH is executed in several parts of algorithm ELTG. In addition, a *random perturbation procedure* is considered to avoid that the algorithm remains in a local minimum for a given number of iterations. Finally, algorithm ELTG calls in sequence procedures *splitting* and *swapping* described in the following subsections.

The main body of algorithm ELTG considers two parts: (1) the construction of an initial solution by using a *Hybrid Initial* procedure, and (2) the *Granular Tabu Search* procedure. The main differences of algorithm ELTG with respect to the algorithm presented in Escobar et al. (2013) for the CLRP are: i) the different granular search space, ii) the linear programming model used in the Hybrid Initial procedure for the assignment of the customers to the depots, iii) the possibility to consider infeasible initial solutions, iv) the procedure *repair* used to remove maximum duration infeasibilities for the initial routes, v) the criteria used to accept a move, vi) the different penalty diversification strategy, and vii) the new local search procedure swapping used within the main loop of the Granular Tabu Search phase.

2.1 Initial solution

The initial MDVRP solution S_0 is constructed by using a *Hybrid Initial* procedure based on a cluster approach, which is able to find good initial solutions within short computing times. The following steps are executed:

– *Step 1*. Construct a giant *Traveling Salesman Problem*(TSP) tour containing all the customers by using the well known *Lin-Kernighan Heuristic* procedure (LKH) as implemented by Helsgaun (2000) (for further details see Lin and Kernighan (1973)).
– *Step 2*. Starting from a given vertex, split the giant TSP tour into several *clusters* (groups of consecutive customers) such that:
  – The number of clusters is not greater than the maximum number of possible routes $M = km$;
  – The total demand of each cluster does not exceed the vehicle capacity Q;
  – The total "duration" dur_g of each cluster g (given by the sum of the service times of its customers and of the costs of the edges connecting consecutive customers) is not greater than $D - \theta \bar{l}$ (where θ is a given parameter, and $\bar{l}$ is the minimum cost of the edges incident to the depots). The value of $\theta \bar{l}$ represents a rough estimation of the cost for connecting the customers of the cluster with a depot.
– *Step 3*. For each depot i and each cluster g, a TSP tour is determined, by using procedure LKH, to obtain the minimum traveling cost (l_{ig}) for visiting the customers belonging to cluster g with a route associated with depot i
– *Step 4*. Assign the depots to the clusters by solving the following Integer Linear Programming (ILP) model, where the binary variable x_{ig} is equal to 1 iff depot i is assigned to cluster g :

$$min\, z = \sum_{i \in I} \sum_{g \in G} l_{ig} x_{ig} + \sigma \sum_{i \in I} \sum_{g \in G} max(0, \bar{d_{ig}} - D) x_{ig} \tag{1}$$

subject to

$$\sum_{i \in I} x_{ig} = 1 \qquad \forall\, g \in G \tag{2}$$

$$\sum_{g \in G} x_{ig} \leq k \qquad \forall\, i \in I \tag{3}$$

$$x_{ig} \in \{0, 1\} \qquad \forall\, i \in I, g \in G \tag{4}$$

where:

$G set\ of\ clusters$

$\sigma penalty\ factor$

$\bar{d}_{ig} = l_{ig} + \sum_{j \in g} \delta_j$, where $\bar{d}_{ig}$ is the duration of cluster$g \in G$ when g is assigned to depot $i \in I$

The objective function (1) sums the traveling costs associated with the edges traversed by the routes and the penalization costs incurred when the maximum duration D is violated. Constraints (2) guarantee that each cluster is assigned to exactly one depot. Constraints (3) guarantee that the number of clusters assigned to each depot does not exceed the number k of vehicles available at each depot.

Constraints (4) can be replaced by $x_{ig} \geq 0, \forall\ i \in I, \forall\ g \in G$, and model (1) - (4) can be rewritten as an equivalent *Linear Programming* (LP) model $Min \left\{c^\top x \mid Ax \leq b \wedge x \geq 0\right\}$. The optimal solutions of both models are equal because matrix A is totally unimodular and b is an integral vector. Indeed, the total unimodularity of matrix A can be proved (see, e.g. Heller and Tompkins (1956)) by considering that:

- every entry in A has value 0 or 1;
- every column of A contains at most two non-zero entries;
- the rows of matrix A can be partitioned into two subsets T_1 and T_2 such that if two non-zero entries in a column of A have the same sign, the row of one of them is in T_1 and the other row is in T_2.

Steps 2–4 are repeated n times, by considering in Step 2 each customer as the possible initial vertex, and keeping the best solution found so far.

As the solution obtained so far can be infeasible with respect to the duration of the routes, the algorithm tries to find a feasible solution by applying a *repair procedure*. This procedure iteratively selects a customer j belonging to an infeasible route and such that the *distance factor* φ_{ij}(where i is the depot to which customer j is currently assigned) is greater than D. Then, customer j is removed from its current route and inserted into a different route (belonging to the same depot or to a different depot) for which the traveling cost c_{jz} ($\forall\ z \in I \cup J$) is minimum.

The proposed algorithm tries to improve the current initial solution by applying a *splitting procedure* based on the procedure proposed by Escobar et al. (2013) for the CLRP. This procedure considers that the total traveling cost can be decreased by adding new routes until the number of routes for each depot is not greater than k, and by assigning them to different depots.

In this procedure, the route which contains the longest edge is selected. Then, its two longest edges, say (r, s) and (t, u), are removed from the route, and the route is shortcut by inserting edge (r, u). The subset of customers belonging to the chain connecting vertex s to vertex t in the considered route is selected as the cluster to form a new route. For each depot i, procedure LKH is applied to find the TSP tour corresponding to the assignment of the cluster to depot i. Each cluster is assigned to the depot for which the cost of the TSP tour is minimum. Then, procedure VRPH is applied to the depots affected by the performed move. The *splitting procedure* is applied N_s times (where N_s is a given parameter), by considering at each iteration a different route. Finally, procedure VRPH is executed for all the depots for which the solution obtained by the *splitting procedure* has not been changed.

One of the key-points for the success of the proposed algorithm is the previously described Hybrid Initial procedure. To evaluate the significant effect of this method, a computational study has been performed to compare the final solutions obtained by executing the proposed algorithm by initially applying the Hybrid Initial procedure and by applying a simple and fast constructive heuristic similar to the approach proposed by Cordeau and Maischberger (2012) for the MDVRP. The corresponding computational experiments are described in Sect. 3. The initialization procedure proposed by Cordeau and Maischberger (2012) performs the following steps:

– Assign each customer to its nearest depot.

For each depot:

– The customers are sorted according to increasing values of the angle they make with the depot and an arbitrary radius.
– Starting with the first route of the considered depot and using this ordering, the customers are inserted one by one into the current route. Whenever the insertion of a customer into the route would lead to a violation of the capacity or of the route duration constraint, a new route is initialized unless there are no more vehicles left for the considered depot. This procedure ensures that the first (m-1) routes of each depot are feasible.

Our simple and fast procedure for finding an alternative initial solution W_0 (called *Alternative Initial* procedure) performs the following steps:

1. Initially, each customer is assigned in turn to its nearest depot in such a way that the total demand of the depot (given by the sum of the demands of the customers already assigned to the depot plus the demand of the considered customer) is not greater than $k * Q$ (global capacity of a depot).
2. For each depot i: Apply the VRP routines *vrp_initial, vrp_sa* and *vrp_rtr* proposed by Groer et al. (2010) by considering all the customers currently assigned to depot i.
3. For each depot i: If the number of routes of depot i is greater than k, then apply the following steps:
   – while the number of routes of depot i is greater than k: remove all the customers from the route containing the smallest number of customers;
   – for each customer h removed from depot i: assign customer h to the nearest depot j (with $j \neq i$) such that the number of routes is not greater than k, and insert customer h into the best position (with respect to the traveling cost) of the route of depot j for which the difference between D and the current duration is maximum.

It is to note that the VRP routines *vrp_initial, vrp_sa* and *vrp_rtr* are not able to control the maximum number of routes generated for each depot, so, at the end of Step 2), the solution corresponding to depot i could be infeasible with respect to this constraint. To avoid this drawback, Step 3) is executed. Indeed, Step 3 allows the transformation of solutions infeasible with respect to the maximum number of routes into solutions (possibly) infeasible with respect to the duration of the routes and/or the capacity of the vehicles. In general, the Granular Tabu Search procedure, described in the next section, is able to repair solutions infeasible with respect to the latter two constraints.

The main difference between our Alternative Initial procedure and the initialization procedure proposed by Cordeau and Maischberger (2012), is that, for each depot, we determine the corresponding VRP solution by using the procedures proposed in Groer et al. (2010), instead of applying a sweep criterion as done in Cordeau and Maischberger (2012).

2.2 Granular Tabu Search procedure

Algorithm ELTG allows solutions which are infeasible with respect to the vehicle capacities and the duration of the routes (see Sect. 2.2.2). The Granular Tabu Search procedure starts by removing the least loaded routes (routes containing one or two customers), and inserting each of the associated customers into the best position, with respect to the objective function $f(S)$ described in Sect. 2.2.2, of one of the remaining routes. In addition, the procedure calls iteratively, during the search, the *splitting* and *swapping procedures*.

The proposed neighborhood structures, the diversification strategies, the intensification strategy, and the *swapping procedure* are described in the following subsections.

2.2.1 Neighborhood structures

The proposed algorithm uses *intra-route* and *inter-route* moves corresponding to the following neighborhood structures:

- *Insertion*. A customer is removed from its current position and reinserted in a different position in the same route or in another route (assigned to the same depot or to a different depot).
- *Swap*. Two customers, belonging to the same route or to different routes (assigned to the same depot or to different depots), are exchanged.
- *Two-opt*. This move is a modified version of the well known two opt move used in solving vehicle routing problems. If the two considered edges are in the same route, the two opt move is equivalent to the intra-route move proposed by Lin and Kernighan (1973) for the TSP. If the two edges are in different routes assigned to the same depot, the move is similar to the traditional inter-route two opt move. The effect of this move becomes more complicated when the edges belong to different depots. In this case, there are several ways to rearrange the routes by performing an additional move concerning the edges connecting the depots with the last customer of the routes to ensure that each route starts and finishes at the same depot.
- *Exchange*. Two consecutive customers are transferred from their current positions to other positions by keeping the edge connecting them. The customers can be inserted in the same route or in a different route (assigned to the same depot or to a different depot).
- *Inter-Swap*. This move is an extension of the Swap move, obtained by considering two pairs of consecutive customers. The edge connecting each pair of customers is kept. The Inter-Swap move is performed between two different routes (assigned to the same depot or to different depots).

A move is performed if at least one of the new edges inserted in the solution belongs to the *sparse graph*. Finally, whenever the algorithm remains in a local minimum for $N_p \times n$ iterations (where N_p is a given parameter), we apply a *random perturbation procedure* which extends the idea of Insertion move by considering three random routes (say r_1, r_2, r_3) at the same time (for further details see Wassan (2005)). In particular, for each customer c_1 of route r_1, each customer c_2 of route r_2, each edge (i_2, j_2) of route r_2 (with $i_2 \neq c_2$ and $j_2 \neq c_2$), and each edge (i_3, j_3) of route r_3, we obtain a new solution S from the best solution found so far by performing the following moves:

- remove customer c_1 from route r_1 and insert it between i_2 and j_2 in route r_2;
- remove customer c_2 from route r_2 and insert it between i_3 and j_3 in route r_3.

The move associated with the solution S corresponding to the minimum value of $c(S) + q(S)$ (see the details in Sect. 2.2.2) is performed, even if solution S is worse than the current solution.

2.2.2 Search, intensification and diversification strategies

The proposed algorithm, as that presented in Gendreau et al. (1994), allows infeasible solutions with respect to both the vehicle capacity and the duration of the routes. Let us consider a solution S composed by a set of z routes $r_1, \ldots, r_z$. Each route r_l where $l \in \{1, \ldots, z\}$ is denoted by $(v_0, v_1, v_2, \ldots, v_0)$. v_0 represents the depot assigned to the route, and $v_1, v_2, \ldots$ represent the visited customers. Let us denote with $v \in r_l$ a customer v belonging to route r_l, and with $(u, v) \in r_l$ an edge such that u and v are two consecutive vertices of route r_l. The following objective function $f(S) = c(S) + \alpha_m \times m(S) + \alpha_q \times q(S)$ is associated with solution S, where:

$$c(S) = \sum_{l=1}^{z} \sum_{(u,v)\in r_l} c_{uv}$$

$$m(S) = \sum_{l=1}^{z} \left[\sum_{v\in r_l} d_v - Q \right]^+$$

$$q(S) = \sum_{l=1}^{z} \left[\left(\sum_{v\in r_l} \delta_v + \sum_{(u,v)\in r_l} c_{uv} \right) - D \right]^+$$

where$[x]^+ = max(0, x)$, and α_m and α_q are two nonnegative weights used to increase the cost of solution S by adding two penalty terms proportional, respectively, to the excess load of the overloaded routes, and to the excess duration of the routes. The values of α_m and α_q are calculated as follows: $\alpha_m = \gamma_m \times f(S_0)$ and $\alpha_q = \gamma_q \times f(S_0)$, where $f(S_0)$ is the value of the objective function of the initial solution S_0, and γ_m and γ_q are two dynamically changing positive parameters adjusted during the search within the range $[\gamma_{min}, \gamma_{max}]$. In particular, if no feasible solutions with respect to the vehicle capacity have been found over N_{mov} iterations, then the value of γ_m is set to $max\{\gamma_{min}, \gamma_m \times r_{pen}\}$, where $r_{pen} < 1$. On the other hand, if feasible solutions with respect to the vehicle capacity have been found during the last N_{mov} iterations, then the value of γ_m is set to $min\{\gamma_{max} \gamma_m \times d_{pen}\}$, where $d_{pen} > 1$. A similar rule is applied to modify the value of γ_q. The initial values of γ_m and γ_q, and the values $\gamma_{min}, \gamma_{max}, N_{mov}, r_{pen}, d_{pen}$ are given parameters.

The proposed algorithm considers three diversification strategies. The first strategy is related to the dynamic modification of the sparse graph proposed by Toth and Vigo (2003). Initially, the sparsification factor β is set to a value β_0. If no improvement of the best solution found so far is obtained during N_β iterations, the subset of edges currently included in the sparse graph is enlarged by increasing the value of β to a value β_n. Then, N_{int} iterations are executed starting from the best solution found so far. Finally, the sparsification factor β is reset to its original value β_0 and the search continues. The values β_0, N_β, β_n and N_{int} are given parameters. It is to note that algorithm ELTG alternates between long intensification phases (small values of β) and short diversification phases (large values of β) allowing the exploration of new parts of the search space.

The second strategy is based on a penalty approach proposed by Taillard (1993). If the considered solution S is feasible, we assign it an objective function value $t(S) = c(S)$. If the solution S is infeasible and the value of the objective function $f(S)$ is less than the cost of the best solution found so far, we assign S a value $t(S) = f(S)$. Otherwise, we add to $f(S)$ an extra penalty term equal to the product of the absolute difference value Δ_{obj} between two successive values of the objective function, the square root of the number of routes z, and a scaling factor h (where h is a given parameter). Therefore, we define $t(S) = f(S) + \Delta_{obj} h \sqrt{z}$. The move corresponding to the minimum value of $t(S)$ is performed. The tabu tenure, as in Gendreau et al. (1994), is randomly selected in the interval $[t_{min}, t_{max}]$ (where t_{min} and t_{max} are given parameters). The following aspiration criterion is used: If the objective function value $f(S)$ of the current solution S is less or equal to the cost of the best solution found so far, solution S is accepted even if it corresponds to a tabu move.

The third diversification strategy considers every $N_{div} \times n$ iterations, the best infeasible solution (i.e. the solution with the smallest value of $c(S)$) and, for each depot, apply procedure VRPH. This strategy helps the algorithm to explore new parts of the solution space. Finally the *splitting procedure* is applied every $N_{split} \times n$ iterations during the Granular Tabu Search phase (where N_{split} is a given parameter).

2.2.3 Swapping procedure

If the traveling costs c_{ij} correspond to euclidean distances, as it is the case for the benchmark MDVRP instances from the literature, the following *swapping procedure* is applied. The procedure starts by selecting the solution S with the smallest value of $c(S)$, and considers the exchange between two depots for a given route r_k. Since each vertex of the input graph G is associated with a point in the plane, route r_k can be represented by its center of gravity (cgr_k). Route r_k is assigned to the depot, say i, different from that currently assigned to route r_k and having the number of routes assigned to it smaller than k, for which the euclidean distance from cgr_k to i is minimum. Procedure VRPH is applied for the two depots involved in the move. If the new solution is feasible and also better than the best solution found so far, the current solution and the best solution found so far are updated; otherwise only the current solution is updated, even if the new solution is worse than the previous one. The swapping procedure is applied every $N_{sw} \times n$ iterations (where N_{sw} is a given parameter).

3 Computational experiments

3.1 Implementation details

Algorithm ELTG has been implemented in C++, and the computational experiments have been performed on an Intel Core Duo (only one core is used) CPU (2.00 GHz) under Linux Ubuntu 11.04 with 2 GB of memory. The LP model equivalent to the ILP model (1)-(4) has been optimally solved by using the LP solver CPLEX 12.1. The performance of algorithm ELTG has been evaluated by considering 33 benchmark instances proposed for the MDVRP. Instances 1-7 were introduced by Christofides and Eilon (1969). Instances 8-11 have been described in Gillett and Johnson (1976). Instances 12-23 were proposed by Chao et al. (1993). Finally, instances 24-33 were introduced by Cordeau et al. (1997). In all the instances, the customers and the depots correspond to random points in the plane. The traveling cost of an edge is calculated as the Euclidean distance between the points corresponding to the extreme vertices of the edge.

Algorithm ELTG has been compared (see Table 2) with the most effective published heuristic algorithms proposed for the MDVRP: Tabu Search (CGL97) of Cordeau et al. (1997), the general heuristic (PR07) of Pisinger and Ropke (2007), the hybrid genetic algorithm (VCGLR12) of Vidal et al. (2012), the sequential tabu search algorithm (CM12) of Cordeau and Maischberger (2012), the hybrid SP+ILS (SUO13) of Subramanian et al. (2013), and the dynamic programming methodology (VCGP14) of Vidal et al. (2014).

For each instance, only one run of algorithm ELTG is executed. Indeed, the computational experiments have shown that the results of the proposed algorithm are independent of the initial value of a “random seed”. The total number of iterations of the main loop of the Granular Tabu Search phase is set to $10 \times n$. The tabu tenure for each move performed is set (as in Gendreau et al. (1994)) to a uniformly distributed random integer number in the interval [5, 10] (i.e., $t_{min} = 5$ and $t_{max} = 10$). As for other metaheuristics, extensive computational tests have been performed to find a suitable set of parameters. On average, the best performance of algorithm ELTG has been obtained by considering the following values of the parameters: $N_{div} = 0.60$, $\theta = 7.0$, $N_s = 3$, $N_p = 0.55$, $\gamma_m = 0.0025$, $\gamma_q = 0.001875$, $\gamma_{min} = \frac{1}{f(S_0)}$, $\gamma_{max} = 0.04$, $N_{mov} = 10$, $r_{pen} = 0.50$, $d_{pen} = 2.00$, $\beta_0 = 1.20$, $N_\beta = 2.50$, $\beta_n = 2.40$, $N_{int} = 1.00$, $h = 0.02$, $N_{split} = 0.70$, and $N_{sw} = 0.90$. These values have been utilized for the solution of all the considered instances.

In Tables 1 and 2, for each instance, the following notation is used:

Instance	Instance number;
n	number of customers;
m	number of depots;
k	maximum number of vehicles available at each depot;
D	maximum duration of each route;
Q	capacity of each vehicle;
Cost	solution cost obtained by the corresponding algorithm in one single run;
BKS	cost of the best-known solution found by the previous algorithms proposed for the MDVRP;
Ref. BKS	reference to the algorithm which obtained for the first time the value BKS;
Best Cost	best solution cost found by the corresponding algorithm over the executed runs;
Avg. Cost	average solution cost found by the corresponding algorithm over the executed runs;
Gap	percentage gap of the solution cost found by the corresponding algorithm in one single run with respect to BKS;
Gap Best	percentage gap of the best solution cost found by the corresponding algorithm over the executed runs with respect to BKS;
Gap Avg.	percentage gap of the average solution cost found by the corresponding algorithm over the executed runs with respect to BKS;
Status	status of the initial solutions W_0 and S_0 obtained by applying, respectively, the Alternative Initial procedure and the Hybrid Initial procedure (*feasible* or *infeasible*);
Time	running time of one single run, expressed in seconds of the CPU used by the corresponding algorithm;
Time Avg.	average running time over the executed runs, expressed in seconds of the CPU used by the corresponding algorithm;
CPU	CPU used by the corresponding algorithm;
CPU index	Passmark performance test for each CPU.

In addition, for each algorithm, the following global values are reported:

Avg.	average percentage gap of the solution cost found by the corresponding algorithm on a subset of instances;
G.Avg	average percentage gap of the solution cost found by the corresponding algorithm on the complete set of instances;
NBKS	number of best solutions (by considering the previous algorithms and algorithm ELTG) found by the corresponding algorithm;
NIBS	number of instances for which the corresponding algorithm is the only one which found the best solution.

For the values of BKS and Ref. BKS, we have considered all the previously published methods proposed for the MDVRP. Therefore, also the results obtained by the exact algorithms and by the heuristic algorithms proposed by Chao et al. (1993)

Table 1 Solutions obtained by each phase of the proposed algorithm by considering initial solutions W_0 and S_0

Characteristics of Instances						BKS	Initial Solution W0				Granular Tabu Search (W0)		
Instance	n	m	k	D	Q		Cost	Gap	Time	Status	Cost	Gap	Time
1	50	4	4	∞	80	**576.87**	609.24	5.61	1	Feasible	**576.87**	0.00	3
2	50	4	2	∞	160	**473.53**	507.01	7.07	1	Feasible	**473.53**	0.00	2
3	75	5	3	∞	140	**641.19**	681.72	6.32	1	Feasible	651.08	1.54	11
4	100	2	8	∞	100	**1001.04**	1016.32	1.53	6	Feasible	1011.03	1.00	32
5	100	2	5	∞	200	**750.03**	774.33	3.24	6	Feasible	760.09	1.34	14
6	100	3	6	∞	100	**876.50**	891.01	1.66	4	Feasible	876.70	0.02	39
7	100	4	4	∞	100	**881.97**	993.65	12.66	3	Feasible	910.21	3.20	31
8	249	2	14	310	500	**4372.78**	4403.54	0.70	28	Feasible	4402.46	0.68	118
9	249	3	12	310	500	**3858.66**	3985.20	3.28	21	Feasible	3931.52	1.89	119
10	249	4	8	310	500	**3631.11**	3825.95	5.37	16	Feasible	3731.80	2.77	107
11	249	5	6	310	500	**3546.06**	3926.93	10.74	13	Feasible	3615.04	1.95	80
12	80	2	5	∞	60	**1318.95**	1365.69	3.54	4	Feasible	**1318.95**	0.00	6
13	80	2	5	200	60	**1318.95**	1365.69	3.54	3	Feasible	**1318.95**	0.00	5
14	80	2	5	180	60	**1360.12**	1365.69	0.41	2	Feasible	1365.69	0.41	4
15	160	4	5	∞	60	**2505.42**	2731.37	9.02	8	Feasible	**2505.42**	0.00	54
16	160	4	5	200	60	**2572.23**	2731.37	6.19	6	Feasible	2575.33	0.12	34
17	160	4	5	180	60	**2709.09**	2731.37	0.82	5	Feasible	2731.37	0.82	33
18	240	6	5	∞	60	**3702.85**	4097.06	10.65	12	Feasible	3824.34	3.28	110
19	240	6	5	200	60	**3827.06**	4097.06	7.06	8	Feasible	3847.56	0.54	86
20	240	6	5	180	60	**4058.07**	4097.06	0.96	7	Feasible	4097.06	0.96	79
21	360	9	5	∞	60	**5474.84**	6145.58	12.25	18	Feasible	5661.76	3.41	109
22	360	9	5	200	60	**5702.16**	6145.58	7.78	12	Feasible	5724.20	0.39	102
23	360	9	5	180	60	**6078.75**	6145.58	1.10	11	Feasible	6145.58	1.10	91
Avg.								**5.28**	**8**			**1.11**	**55**

Table 1 continued

Characteristics of Instances						BKS	Initial Solution W0				Granular Tabu Search (W0)		
Instance	n	m	k	D	Q		Cost	Gap	Time	Status	Cost	Gap	Time
24	48	4	1	500	200	**861.32**	997.17	15.77	1	Feasible	**861.32**	0.00	2
25	96	4	2	480	195	**1307.34**	1524.48	16.61	3	Feasible	1312.60	0.40	6
26	144	4	3	460	190	**1803.80**	1909.40	5.85	6	Feasible	1866.34	3.47	43
27	192	4	4	440	185	**2058.31**	2393.07	16.26	10	Feasible	2126.08	3.29	42
28	240	4	5	420	180	**2331.20**	2875.69	23.36	15	Infeasible	2452.49	5.20	69
29	288	4	6	400	175	**2676.30**	3126.97	16.84	20	Feasible	2767.98	3.43	98
30	72	6	1	500	200	**1089.56**	1396.17	28.14	1	Infeasible	**1089.56**	0.00	4
31	144	6	2	475	190	**1664.85**	2045.70	22.88	4	Feasible	1722.78	3.48	20
32	216	6	3	450	180	**2133.20**	2276.17	6.70	9	Feasible	2159.26	1.22	69
33	288	6	4	425	170	**2868.20**	3771.05	31.48	17	Infeasible	3100.50	8.10	110
Avg.								**18.39**	**9**			**2.86**	**46**
G. Avg								**9.25**	**8**			**1.64**	**52**

Table 1 continued

Characteristics of Instances						Initial Solution S0				Granular Tabu Search (S0)		
Instance	n	m	k	D	Q	Cost	Gap	Time	Status	Cost	Gap	Time
1	50	4	4	∞	80	594.52	3.06	5	Feasible	**576.87**	0.00	7
2	50	4	2	∞	160	492.18	3.94	4	Feasible	**473.53**	0.00	6
3	75	5	3	∞	140	693.25	8.12	19	Feasible	**641.19**	0.00	29
4	100	2	8	∞	100	1018.47	1.74	62	Feasible	**1001.04**	0.00	90
5	100	2	5	∞	200	751.26	0.16	17	Feasible	**750.03**	0.00	26
6	100	3	6	∞	100	918.29	4.77	65	Feasible	**876.50**	0.00	103
7	100	4	4	∞	100	945.00	7.15	76	Feasible	884.66	0.30	106
8	249	2	14	310	500	4584.97	4.85	185	Feasible	**4371.66**	−0.03	285
9	249	3	12	310	500	4009.69	3.91	156	Feasible	3880.85	0.58	256
10	249	4	8	310	500	3854.68	6.16	166	Feasible	**3629.60**	−0.04	267
11	249	5	6	310	500	3738.33	5.42	125	Feasible	**3545.18**	−0.02	192
12	80	2	5	∞	60	1369.47	3.83	4	Feasible	**1318.95**	0.00	6
13	80	2	5	200	60	1349.07	2.28	5	Feasible	**1318.95**	0.00	7
14	80	2	5	180	60	**1360.12**	0.00	4	Feasible	**1360.12**	0.00	6
15	160	4	5	∞	60	2590.87	3.41	69	Feasible	**2505.42**	0.00	114
16	160	4	5	200	60	2761.25	7.35	87	Feasible	**2572.23**	0.00	118
17	160	4	5	180	60	2895.76	6.89	79	Feasible	**2709.09**	0.00	108
18	240	6	5	∞	60	4111.78	11.04	178	Feasible	**3702.85**	0.00	278
19	240	6	5	200	60	4292.11	12.15	176	Feasible	**3827.06**	0.00	256
20	240	6	5	180	60	4441.59	9.45	190	Infeasible	**4058.07**	0.00	267
21	360	9	5	∞	60	6106.37	11.54	166	Feasible	**5474.84**	0.00	268
22	360	9	5	200	60	6613.80	15.99	170	Infeasible	**5702.16**	0.00	262
23	360	9	5	180	60	6677.53	9.85	199	Infeasible	6095.46	0.27	285
Avg.							**6.22**	**96**			**0.05**	**145**

Table 1 continued

Characteristics of Instances						Initial Solution S0				Granular Tabu Search (S0)		
Instance	n	m	k	D	Q	Cost	Gap	Time	Status	Cost	Gap	Time
24	48	4	1	500	200	894.26	3.82	2	Feasible	**861.32**	0.00	4
25	96	4	2	480	195	1449.20	10.85	8	Infeasible	1311.11	0.29	11
26	144	4	3	460	190	1883.80	4.44	72	Feasible	**1803.80**	0.00	118
27	192	4	4	440	185	2103.46	2.19	89	Feasible	2064.11	0.28	124
28	240	4	5	420	180	2466.38	5.80	147	Feasible	2349.63	0.79	213
29	288	4	6	400	175	2769.73	3.49	145	Feasible	2710.30	1.27	234
30	72	6	1	500	200	1255.87	15.26	8	Feasible	**1089.56**	0.00	11
31	144	6	2	475	190	1883.39	13.13	47	Feasible	1665.50	0.04	66
32	216	6	3	450	180	2258.09	5.85	94	Feasible	2151.45	0.86	156
33	288	6	4	425	170	3046.00	6.20	199	Feasible	2910.78	1.48	302
Avg.							**7.10**	**81**			**0.50**	**124**
G. Avg							**6.49**	**91**			**0.18**	**139**

The costs which are equal to the corresponding value of BKS are reported in bold. Whenever algorithm ELTG improves the BKS value, the reported cost is underlined

Table 2 Solutions (CPU Times) obtained by the MDVRP Algorithms

Instance	Previous Solutions		CGL97 (1 run)			PR07 (10 runs)				
	BKS	Ref. BKS	Cost	Gap	Time	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.
1	**576.87**	CGW93	**576.87**	0.00	194	**576.87**	0.00	**576.87**	0.00	29
2	**473.53**	RLB96	**473.87**	0.07	208	**473.53**	0.00	**473.53**	0.00	28
3	**641.19**	PR07	645.15	0.62	340	**641.19**	0.00	**641.19**	0.00	64
4	**1001.04**	PR07	1006.66	0.56	467	**1001.04**	0.00	1006.09	0.50	88
5	**750.03**	CGL97	753.34	0.44	493	751.26	0.16	752.34	0.31	120
6	**876.50**	RLB96	877.84	0.15	459	876.70	0.02	883.01	0.74	93
7	**881.97**	PR07	891.95	1.13	463	**881.97**	0.00	889.36	0.84	88
8	**4372.78**	VCGLR12	4482.44	2.51	1526	4390.80	0.41	4421.03	1.10	333
9	**3858.66**	VCGLR12	3920.85	1.61	1604	3873.64	0.39	3892.50	0.88	361
10	**3631.11**	VCGLR12	3714.65	2.30	1530	3650.04	0.52	3666.85	0.98	363
11	**3546.06**	PR07	3580.84	0.98	1555	3546.06	0.00	3573.23	0.77	357
12	**1318.95**	RLB96	**1318.95**	0.00	334	**1318.95**	0.00	1319.13	0.01	75
13	**1318.95**	RLB96	**1318.95**	0.00	335	**1318.95**	0.00	**1318.95**	0.00	60
14	**1360.12**	CGL97	**1360.12**	0.00	326	**1360.12**	0.00	**1360.12**	0.00	58
15	**2505.42**	CGL97	2534.13	1.15	844	**2505.42**	0.00	2519.64	0.57	253
16	**2572.23**	RLB96	**2572.23**	0.00	843	**2572.23**	0.00	2573.95	0.07	188
17	**2709.09**	CGL97	2720.23	0.41	822	**2709.09**	0.00	**2709.09**	0.00	179
18	**3702.85**	CGL97	3710.49	0.21	1491	**3702.85**	0.00	3736.53	0.91	419
19	**3827.06**	RLB96	**3827.06**	0.00	1512	**3827.06**	0.00	3838.76	0.31	315
20	**4058.07**	CGL97	**4058.07**	0.00	1483	**4058.07**	0.00	4064.76	0.16	300
21	**5474.84**	CGL97	5535.99	1.12	2890	**5474.84**	0.00	5501.58	0.49	582
22	**5702.16**	CGL97	5716.01	0.24	2934	**5702.16**	0.00	5722.19	0.35	462
23	**6078.75**	PR07	6139.73	1.00	2872	**6078.75**	0.00	6092.66	0.23	443
Avg.				**0.63**	**1110**		**0.07**		**0.40**	**229**

Table 2 continued

Instance	Previous Solutions		CGL97 (1 run)			PR07 (10 runs)				
	BKS	Ref. BKS	Cost	Gap	Time	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.
24	**861.32**	CGL97	**861.32**	0.00	242	**861.32**	0.00	**861.32**	0.00	30
25	**1307.34**	PR07	1314.99	0.59	505	**1307.34**	0.00	1308.17	0.06	103
26	**1803.80**	VCGLR12	1815.62	0.66	854	1806.00	0.12	1810.66	0.38	214
27	**2058.31**	VCGLR12	2094.24	1.75	1158	2060.93	0.13	2073.16	0.72	296
28	**2331.20**	VCGLR12	2408.10	3.30	1529	2337.84	0.28	2350.31	0.82	372
29	**2676.30**	VCGLR12	2768.13	3.43	2007	2687.60	0.42	2695.74	0.73	465
30	**1089.56**	PR07	1092.12	0.23	412	**1089.56**	0.00	**1089.56**	0.00	58
31	**1664.85**	PR07	1676.26	0.69	906	**1664.85**	0.00	1675.74	0.65	207
32	**2133.20**	VCGLR12	2176.79	2.04	1462	2136.42	0.15	2144.84	0.55	350
33	**2868.20**	VCGLR12	3089.62	7.72	2105	2889.82	0.75	2905.43	1.30	455
Avg.				**2.04**	**1118**		**0.19**		**0.52**	**255**
G. Avg				**1.06**	**1112**		**0.10**		**0.44**	**237**
NBKS			**8**			**22**		**8**		
NIBS			**0**			**0**		**0**		
CPU			**Sun Sparcstation 10**			**Pentium 4 (3.0 GHz)**				
CPU index			**---**			**489**				

Table 2 continued

Instance	VCGLR12 (10 runs)					CM12 (10 runs)				
	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.
1	**576.87**	0.00	**576.87**	0.00	14	**576.87**	0.00	**576.87**	0.00	-
2	**473.53**	0.00	**473.53**	0.00	13	**473.53**	0.00	**473.53**	0.00	-
3	**641.19**	0.00	**641.19**	0.00	26	**641.19**	0.00	**641.19**	0.00	-
4	**1001.04**	0.00	1001.23	0.02	116	**1001.04**	0.00	1002.64	0.16	-
5	**750.03**	0.00	**750.03**	0.00	64	**750.03**	0.00	750.41	0.05	-
6	**876.50**	0.00	**876.50**	0.00	68	**876.50**	0.00	877.03	0.06	-
7	**881.97**	0.00	884.43	0.28	93	**881.97**	0.00	884.18	0.25	-
8	4372.78	0.00	4397.42	0.56	600	4409.35	0.84	4438.47	1.50	-
9	**3858.66**	0.00	3868.59	0.26	570	3881.16	0.58	3894.10	0.92	-
10	3631.11	0.00	3636.08	0.14	589	3633.88	0.08	3660.39	0.81	-
11	3546.06	0.00	3548.25	0.06	428	3548.09	0.06	3553.88	0.22	-
12	**1318.95**	0.00	**1318.95**	0.00	31	**1318.95**	0.00	**1318.95**	0.00	-
13	**1318.95**	0.00	**1318.95**	0.00	34	**1318.95**	0.00	**1318.95**	0.00	-
14	**1360.12**	0.00	**1360.12**	0.00	33	**1360.12**	0.00	**1360.12**	0.00	-
15	**2505.42**	0.00	**2505.42**	0.00	115	**2505.42**	0.00	**2505.42**	0.00	-
16	**2572.23**	0.00	**2572.23**	0.00	118	**2572.23**	0.00	**2572.23**	0.00	-
17	**2709.09**	0.00	**2709.09**	0.00	128	**2709.09**	0.00	**2709.09**	0.00	-
18	**3702.85**	0.00	**3702.85**	0.00	271	**3702.85**	0.00	3703.96	0.03	-
19	**3827.06**	0.00	**3827.06**	0.00	252	**3827.06**	0.00	**3827.06**	0.00	-
20	**4058.07**	0.00	**4058.07**	0.00	262	**4058.07**	0.00	**4058.07**	0.00	-
21	**5474.84**	0.00	5476.41	0.03	600	**5474.84**	0.00	5486.91	0.22	-
22	**5702.16**	0.00	**5702.16**	0.00	600	**5702.16**	0.00	5708.44	0.11	-
23	**6078.75**	0.00	**6078.75**	0.00	600	**6078.75**	0.00	6086.05	0.12	-
Avg.		**0.00**		**0.06**	**245**		**0.07**		**0.19**	**-**

Table 2 continued

Instance	VCGLR12 (10 runs)					CM12 (10 runs)				
	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.
24	**861.32**	0.00	**861.32**	0.00	10	**861.32**	0.00	**861.32**	0.00	-
25	**1307.34**	0.00	**1307.34**	0.00	46	**1307.34**	0.00	1307.73	0.03	-
26	**1803.80**	0.00	**1803.80**	0.00	115	1804.36	0.03	1805.24	0.08	-
27	**2058.31**	0.00	2059.36	0.05	313	2058.47	0.01	2071.48	0.64	-
28	**2331.20**	0.00	2340.29	0.39	574	2340.31	0.39	2355.44	1.04	-
29	**2676.30**	0.00	2681.93	0.21	600	2688.54	0.46	2699.58	0.87	-
30	**1089.56**	0.00	**1089.56**	0.00	20	**1089.56**	0.00	**1089.56**	0.00	-
31	**1664.85**	0.00	1665.05	0.01	123	**1664.85**	0.00	1666.85	0.12	-
32	**2133.20**	0.00	2134.17	0.05	366	2141.45	0.39	2150.48	0.81	-
33	**2868.20**	0.00	2886.59	0.64	600	2887.05	0.66	2911.86	1.52	-
Avg.		**0.00**		**0.14**	**277**		**0.19**		**0.51**	**-**
G. Avg		**0.00**		**0.08**	**254**		**0.11**		**0.29**	**1101**
NBKS	**30**		**20**			**23**		**13**		
NIBS	**3**		**0**			**0**		**0**		
CPU	**AMD Opteron 250 (2.4 GHz)**					**Xeon X7350 (2.93 Ghz)**				
CPU index	**1411**					**16715**				

Table 2 continued

Instance	SUO13 (10 runs)					VCGP14 (10 runs)					ELTG (1 run)		
	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.	Cost Cost	Gap Gap	Time Time
1	**576.87**	0.00	**576.87**	0.00	3	**576.87**	0.00	**576.87**	0.00	65	**576.87**	0.00	7
2	**473.53**	0.00	**473.53**	0.00	2	**473.53**	0.00	**473.53**	0.00	101	**473.53**	0.00	6
3	**641.19**	0.00	**641.19**	0.00	7	-	-	-	-	-	**641.19**	0.00	29
4	**1001.04**	0.00	**1001.04**	0.00	52	-	-	-	-	-	**1001.04**	0.00	90
5	**750.03**	0.00	750.21	0.02	32	**750.03**	0.00	**750.03**	0.00	293	**750.03**	0.00	26
6	**876.50**	0.00	**876.50**	0.00	26	**876.50**	0.00	**876.50**	0.00	181	**876.50**	0.00	103
7	**881.97**	0.00	**881.97**	0.00	22	**881.97**	0.00	**881.97**	0.00	225	884.66	0.30	106
8	4379.46	0.15	4393.70	0.48	1245	4375.49	0.06	4383.63	0.25	1196	**4371.66**	−0.03	285
9	3859.54	0.02	3864.22	0.14	1432	3859.17	0.01	3860.77	0.05	1171	3880.85	0.58	256
10	3631.37	0.01	3634.72	0.10	1423	3631.11	0.00	3631.71	0.02	1063	**3629.60**	−0.04	267
11	3546.06	0.00	3546.15	0.00	1217	3546.06	0.00	3547.37	0.04	1028	**3545.18**	−0.02	192
12	**1318.95**	0.00	**1318.95**	0.00	6	**1318.95**	0.00	**1318.95**	0.00	171	**1318.95**	0.00	6
13	**1318.95**	0.00	**1318.95**	0.00	3	**1318.95**	0.00	**1318.95**	0.00	171	**1318.95**	0.00	7
14	**1360.12**	0.00	**1360.12**	0.00	19	**1360.12**	0.00	**1360.12**	0.00	152	**1360.12**	0.00	6
15	**2505.42**	0.00	**2505.42**	0.00	49	**2505.42**	0.00	**2505.42**	0.00	467	**2505.42**	0.00	114
16	**2572.23**	0.00	**2572.23**	0.00	248	**2572.23**	0.00	**2572.23**	0.00	465	**2572.23**	0.00	118
17	**2709.09**	0.00	2710.21	0.04	1448	**2709.09**	0.00	**2709.09**	0.00	499	**2709.09**	0.00	108
18	**3702.85**	0.00	**3702.85**	0.00	1020	**3702.85**	0.00	**3702.85**	0.00	841	**3702.85**	0.00	278
19	**3827.06**	0.00	3827.55	0.01	1215	**3827.06**	0.00	**3827.06**	0.00	907	**3827.06**	0.00	256
20	**4058.07**	0.00	**4058.07**	0.00	545	**4058.07**	0.00	**4058.07**	0.00	972	**4058.07**	0.00	267
21	**5474.84**	0.00	**5474.84**	0.00	2545	**5474.84**	0.00	**5474.84**	0.00	1204	**5474.84**	0.00	268
22	**5702.16**	0.00	5705.84	0.06	846	**5702.16**	0.00	**5702.16**	0.00	1200	**5702.16**	0.00	262
23	**6078.75**	0.00	**6078.75**	0.00	1019	**6078.75**	0.00	6080.43	0.03	1200	6095.46	0.27	285
Avg.		**0.01**		**0.04**	**627**		**-**		**-**	**-**		**0.05**	**145**

Table 2 continued

Instance	SUO13 (10 runs)					VCGP14 (10 runs)					ELTG (1 run)		
	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.	Best Cost	Gap Best	Avg. Cost	Gap Avg.	Time Avg.	Cost Cost	Gap Gap	Time Time
24	**861.32**	0.00	**861.32**	0.00	1	**861.32**	0.00	**861.32**	0.00	123	**861.32**	0.00	4
25	**1307.34**	0.00	1308.53	0.09	12	-	-	-	-	-	1311.11	0.29	11
26	**1803.80**	0.00	1804.09	0.02	55	**1803.80**	0.00	**1803.80**	0.00	530	**1803.80**	0.00	118
27	**2058.31**	0.00	2060.93	0.13	779	-	-	-	-	-	2064.11	0.28	124
28	**2331.20**	0.00	2338.12	0.30	1337	-	-	-	-	-	2349.63	0.79	213
29	2680.77	0.17	2685.23	0.33	2298	-	-	-	-	-	2710.30	1.27	234
30	**1089.56**	0.00	**1089.56**	0.00	4	-	-	-	-	-	**1089.56**	0.00	11
31	**1664.85**	0.00	1665.08	0.01	394	-	-	-	-	-	1665.50	0.04	66
32	**2133.20**	0.00	2135.37	0.10	1070	-	-	-	-	-	2151.45	0.86	156
33	2874.28	0.21	2882.41	0.50	3010	-	-	-	-	-	2910.78	1.48	302
Avg.		**0.04**		**0.15**	896		**-**		**-**	**-**		**0.50**	**124**
G. Avg		**0.02**		**0.07**	709		**-**		**-**	**-**		**0.18**	**139**
NBKS	**27**		**17**			**-**		**-**			**23**		
NIBS	**0**		**0**			**-**		**-**			**3**		
CPU	**Intel Core i7 (2.93 GHz)**					**AMD Opteron 250 (2.4 GHz)**					**Core Duo (2.0 GHz)**		
CPU index	**5454**					**1411**					**1398**		

The costs which are equal to the corresponding value of BKS are reported in bold. Whenever algorithm ELTG improves the BKS value, the reported cost is underlined

(CGW93) and by Renaud et al. (1996a) (RLB96), have been taken into account. The optimality of the value of BKS has been proved for instances 1, 2, 6, 7 and 12 by Baldacci and Mingozzi (2009), and for instances 3, 4, 5, 13 to 20, 24, 26, 30 and 31 by Contardo and Martinelli (2014). For each instance, the costs which are equal to the corresponding value of BKS are reported in bold. Whenever algorithm ELTG improves the BKS value, the reported cost is underlined. The CPU index is given by the Passmark performance test (for further details see PassMark (2012)). This is a well known benchmark test focused on CPU and memory performance. Higher values of the Passmark test indicate that the corresponding CPU is faster. Note that for the CPU used for algorithm CGL97, the value of the CPU index is not available (this CPU is however much slower than those used for the other algorithms).

3.2 Global results

Table 1 provides the results obtained by the Alternative Initial procedure (solution W_0), the Hybrid Initial procedure (solution S_0), and the Granular Tabu Search procedure of algorithm ELTG by considering as starting solution W_0 and S_0, respectively. The table shows, for each instance, the results (cost, value of Gap BKS and cumulative running time) corresponding to the following solutions:

- Initial Solution W_0: solution obtained after the application of the Alternative Initial procedure;
- Initial Solution S_0: solution obtained after the application of the Hybrid Initial procedure;
- Granular Tabu Search (W_0): solution obtained (at the end of the Granular Tabu Search procedure) by algorithm ELTG by considering the initial solution W_0;
- Granular Tabu Search (S_0): solution obtained (at the end of the Granular Tabu Search procedure) by algorithm ELTG by considering the initial solution S_0.

Whenever an initial solution W_0 or S_0 is infeasible with respect to the duration and capacity of the routes, or to the number of routes for each depot, its status is set to *infeasible*. Otherwise, its status is set to *feasible*. It is to note that the Granular Tabu Search procedure produces substantial improvements, within short additional running times, on all the instances, but instances 14, 17, 20 and 23 when starting from solution W_0.

Table 1 shows that the Alternative Initial procedure is faster than the Hybrid Initial procedure, and produces, for the first 23 instances, some initial solutions W_0 better than the corresponding initial solutions S_0, with a smaller average value of Gap BKS. It is to note however that the global better behavior of the former procedure on these 23 instances is mainly due to its performance on the subset defined by the last 12 instances (i.e. instances 12, ..., 23), which are "structured", "symmetric" and not tightly constrained instances, such that the assignment of each customer to its closest depot (as done in the first step of the alternative initial procedure) generally represents a feasible and proper assignment. If we consider, as an example, instances 18, 19 and 20 (having the same input data but the values of D), the three corresponding initial solutions W_0 are identical: 4 routes for each of the 6 depots, each route with a global demand equal to 54 and a cost equal to 170.71. Although for some instances the initial solutions W_0 are better than the corresponding initial solutions S_0, the final solutions obtained after the execution of the Granular Tabu Search procedure are always better (sometimes much better) by starting from solutions S_0 than from solutions W_0. As a consequence, in the proposed algorithm ELTG, we will apply the Granular Tabu Search procedure after the execution of the Hybrid Initial procedure.

A summary on the results obtained by the considered algorithms (CGL97, PR07, VCGLR12, CM12, SUO13, VCGP14 and ELTG) for the complete set of instances is given in Table 2. In this table we report the results as presented in the corresponding papers. As said in the Introduction, algorithm VCGP14 considers the relaxed case of the MDVRP where no constraint on the number of routes associated with each depot is imposed. This constraint is not tight (i.e. it does not affect the feasibility of the solutions found) for instances 1, 2, 5 to 24, and 26, while, for the remaining instances, the solutions found by algorithm VCGP14 are infeasible for the classical version of the MDVRP. For this reason, in Table 2 we report only the results found by algorithm VCGP14 on the instances for which a feasible solution was found. Algorithms PR07, VCGLR12, SUO13, and VCGP14 have been executed for ten runs. The results reported for these algorithms correspond, for each instance, to the best and to the average costs found, and to the average CPU time over the ten runs. For algorithm CM12, the results reported correspond, for each instance, to the best and to the average costs found and to the average CPU time obtained over ten runs, with 10^6 iterations for each run. Finally, the results reported for algorithms CGL97 and ELTG correspond, for each instance, to a single run of the corresponding algorithm. For what concerns a comparison among the reported CPU times, it is necessary to take into account the different speeds of the CPUs used in the computational experiments. In addition, for the algorithms reporting average values of the CPU times, i.e. algorithms PR07, VCGLR12, CM12, SUO13, and VCGP14, which execute ten runs for each instance, the CPU times corresponding to the best found costs should be multiplied times the number of executed runs.

On the first 23 instances, Table 2 shows that algorithm ELTG provides a value of the global average value of Gap BKS better than that of algorithms CGL97, PR07, and CM12. In addition, by considering the global average value of the gaps corresponding to the average costs computed over several runs (Gap Avg.) on these instances, Table 2 shows that algorithm ELTG obtains results better than those obtained (in slightly larger CPU times) by algorithm VCGLR12, and results similar to those obtained (in much larger CPU times) by algorithm SUO13. The best results on the global average value of Gap Best are obtained, with very large CPU times, by algorithms VCGLR12 and SUO13. By taking into account the big difference of the corresponding CPU times, it is difficult to make a direct comparison of the quality of the solutions found by algorithm ELTG with respect to the best results reported for algorithms VCGLR12 and SUO13.

For instances 24 - 33, algorithm ELTG has a global average value of Gap Avg. smaller than that of algorithms CGL97, PR07, and CM12; only algorithms VCGLR12 and SUO13 provide, although with longer CPU times, a better global average value of Gap Avg.

By considering the complete data set, algorithm ELTG obtains excellent results for what concerns the number (NBKS) of best known solutions found and the number

(NIBS) of instances for which the corresponding algorithm is the only one which finds the best known solution. Indeed, it is able to find, within short CPU times, 20 best known solutions, and to improve the previous best known solution for 3 instances which have been deeply studied in the past years. It is also to note that only algorithms VCGLR12 (by considering the best costs found) and ELTG are able to find, each for three instances, solutions strictly better than those found by all the other algorithms.

As for the average CPU time, algorithm ELTG is much faster than algorithms PR07 (by considering the best costs found), VCGLR12, CM12, and SUO13, which are the only ones able to find better results in terms of the average value of Gap Best. The average running time of algorithm ELTG is larger than that of algorithms CGL97 and PR07 (by considering the average costs found), which, on the other hand, find solutions worse than those found by algorithm ELTG.

Note that the average and best results on the complete data set are not available for algorithm VCGP14, therefore a global comparison with this algorithm cannot be performed. By comparing algorithms ELTG and VCGP14 on the 23 instances for which the latter algorithm finds feasible solutions, we can note that algorithm ELTG finds 20 best solutions (of which 3 are new), while algorithm VCGP14 finds (in much larger CPU times) 19 best solutions.

4 Concluding remarks

We propose an effective hybrid Granular Tabu Search algorithm for the Multi Depot Vehicle Routing Problem (MDVRP). The algorithm is based on the heuristic framework introduced by Escobar et al. (2013) for the *Capacitated Location Routing Problem* (CLRP). In the proposed approach, after the construction of an initial solution by using a hybrid heuristic, we apply a modified Granular Tabu Search procedure which considers five granular neighborhoods, three different diversification strategies and different local search procedures. A perturbation procedure is applied whenever the algorithm remains in a local optimum for a given number of iterations.

We compare the proposed algorithm with the most effective published heuristics for the MDVRP on a set of benchmark instances from the literature. The results show the effectiveness of the proposed algorithm, and some best known solutions are improved within reasonable computing times. The results obtained suggest that the proposed framework could be applied to other extensions of the MDVRP such as the Multi Depot Periodic Vehicle Routing Problem (MDPVRP), the Multi Depot Periodic Vehicle Routing Problem with Heterogeneous Fleet (HMDVRP), and other problems obtained by adding constraints as time windows, pickups and deliveries, etc.

Acknowledgments The work of the first author has been partially supported by MIUR (Ministero Istruzione, Università e Ricerca), Italy, and Pontificia Universidad Javeriana Cali, Colombia. This support is gratefully acknowledged.

References

Baldacci, R., Mingozzi, A.: A unified exact method for solving different classes of vehicle routing problems. Math. Program. **120**(2), 347–380 (2009)
Chao, I., Golden, B., Wasil, E.: A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. Am. J. Math. Manag. Sci. **13**(3), 371–406 (1993)
Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. Oper. Res. Q. **20**(3), 309–318 (1969)
Contardo, C., Martinelli, R.: A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Discret. Optim. **12**, 129–146 (2014)
Cordeau, J., Maischberger, M.: A parallel iterated tabu search heuristic for vehicle routing problems. Comput. Oper. Res. **39**(9), 2033–2050 (2012)
Cordeau, J., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks **30**(2), 105–119 (1997)
Dueck, G.: New optimization heuristics. J. Comput. Phys. **104**(1), 86–92 (1993)
Escobar, J., Linfati, R., Toth, P.: A two-phase hybrid metaheuristic algorithm for the capacitated location-routing problem. Comput. Oper. Res **40**(1), 70–79 (2013)
Gendreau, M., Hertz, A., Laporte, G.: New insertion and postoptimization procedures for the traveling salesman problem. Oper. Res. **40**(6), 1086–1094 (1992)
Gendreau, M., Hertz, A., Laporte, G.: A tabu search heuristic for the vehicle routing problem. Manage. Sci. **40**(10), 1276–1290 (1994)
Gillett, B., Johnson, J.: Multi-terminal vehicle-dispatch algorithm. Omega **4**(6), 711–718 (1976)
Gillett, B., Miller, L.: A heuristic algorithm for the vehicle-dispatch problem. Oper. Res. **22**(2), 340–349 (1974)
Golden, B., Magnanti, T., Nguyen, H.: Implementing vehicle routing algorithms. Networks **7**(2), 113–148 (1977)
Groer, C., Golden, B., Wasil, E.: A library of local search heuristics for the vehicle routing problem. Math. Program. Comput. **2**(2), 79–101 (2010)
Gulczynski, D., Golden, B., Savelsbergh, M., Wasil, E.: The multi-depot vehicle routing problem: an integer programming-based heuristic and computational results. In: Siarry, P. (ed.) Heuristics: Theory and Applications, pp. 287–309. Nova Science Publishers Inc, New York (2013)
Heller, I., Tompkins, C.: An extension of a theorem of dantzig. Ann. Math. Stud. **38**, 247–254 (1956)
Helsgaun, K.: An effective implementation of the lin-kernighan traveling salesman heuristic. Eur. J. Oper. Res. **126**(1), 106–130 (2000)
Laporte, G., Nobert, Y., Arpin, D.: Optimal solutions to capacitated multi-depot vehicle routing problems. Congressus Numerantium **44**, 283–292 (1984)
Laporte, G., Nobert, Y., Taillefer, S.: Solving a family of multi-depot vehicle routing and location-routing problems. Trans. Sci. **22**(3), 161–172 (1988)
Lin, S., Kernighan, B.: An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. **21**(2), 498–516 (1973)
Ombuki-Berman, B., Hanshar, F.: Using genetic algorithms for multi-depot vehicle routing. Stud. Comput. Intell. **161**, 77–99 (2009)
PassMark: PassMark Performance Test. http://www.passmark.com (2012). Accessed 14-May-2012
Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Comput. Oper. Res. **34**(8), 2403–2435 (2007)
Raft, O.: A modular algorithm for an extended vehicle scheduling problem. Eur. J. Oper. Res. **11**(1), 67–76 (1982)
Renaud, J., Boctor, F., Laporte, G.: An improved petal heuristic for the vehicle routing problem. J. Oper. Res. Soc. **47**(2), 329–336 (1996a)
Renaud, J., Laporte, G., Boctor, F.: A tabu search heuristic for the multi-depot vehicle routing problem. Computers and Operations Research **23**(3), 229–235 (1996b)
Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G.: Record breaking optimization results using the ruin and recreate principle. J. Comput. Phys. **159**(2), 139–171 (2000)
Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. Princ. Pract. Constraint Program. **1520**, 417–431 (1998)
Subramanian, A., Uchoa, E., Ochi, L.S.: A hybrid algorithm for a class of vehicle routing problems. Comput. Oper. Res. **40**(10), 2519–2531 (2013)

Taillard, É.: Parallel iterative search methods for vehicle routing problems. Networks **23**(8), 661–673 (1993)

Thangiah, S., Salhi, S.: Genetic clustering: an adaptive heuristic for the multidepot vehicle routing problem. Appl. Artif. Intell. **15**(4), 361–383 (2001)

Toth, P., Vigo, D.: The granular tabu search and its application to the vehicle-routing problem. INF. J. Comput. **15**(4), 333–346 (2003)

Vidal, T., Crainic, T., Gendreau, M., Lahrichi, N., Rei, W.: A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. Oper. Res. **60**(3), 611–624 (2012)

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: Implicit depot assignments and rotations in vehicle routing heuristics. Eur. J. Oper. Res. (2014). doi:10.1016/jejor201312044

Wassan, N.: A reactive tabu search for the vehicle routing problem. J. Oper. Res. Soc. **57**(1), 111–116 (2005)

Wren, A., Holliday, A.: Computer scheduling of vehicles from one or more depots to a number of delivery points. Oper. Res. Q. **23**(3), 333–344 (1972)