Alma Mater Studiorum – Università di Bologna

Dottorato di Ricerca in
Automatica e Ricerca Operativa

Ciclo XXV
Settore Concorsuale di afferenza: 01/A6
Settore Scientifico disciplinare: MAT/09

Heuristic algorithms for the Capacitated Location-Routing Problem and the Multi-Depot Vehicle Routing Problem

Presentata da: John Willmer Escobar Velasquez

Coordinatore Dottorato
Prof. Andrea Lodi

Relatore
Prof. Paolo Toth

Esame finale anno 2013

To my parents: Maria Isabel and Jaime

and in loving memory of my grandparents

Acknowledgments

First and foremost, I have to thank God for all his love. I would love to express my sincere gratitude to my Advisor Prof. Paolo Toth for giving me the wonderful opportunity to work with him. Many thanks to him for his teachings, his helpful suggestions and remarks, his advice and his patience. Indeed, this work would not have been possible without him.

I want also to thank to Prof. Maria Gulnara Baldoquin and Rodrigo Linfati. I would like to thank to Prof. Gulnara who acted as co-advisor of this work. Many thanks to Rodrigo for his invaluable friendship and support. In addition, I wish also to thank to my friend "Mauro" with whom I shared great experiences in Bologna.

I also wish to express my gratitude with all the members of the Operations Research group of Bologna, for their professional help and their friendship.

I want to thank my parents, for supporting me always. Many thanks to them for their endless love, encouragement and motivation throughout my life. I also wish to express my gratitude to the love of my life and wife, Andrea. This dream would not have been possible without them.

Finally, this work has been partially supported by MIUR (Ministero Istruzione, Università e Ricerca), Italy, and Pontificia Universidad Javeriana Cali, Colombia. This support is gratefully acknowledged.

Bologna, March 15, 2013 John Willmer Escobar V.

Abstract

The Capacitated Location-Routing Problem (CLRP) is a NP-hard problem since it generalizes two well known NP-hard problems: the Capacitated Facility Location Problem (CFLP) and the Capacitated Vehicle Routing Problem (CVRP). The Multi-Depot Vehicle Routing Problem (MDVRP) is known to be a NP-hard since it is a generalization of the well known Vehicle Routing Problem (VRP), arising with one depot. This thesis addresses heuristics algorithms based on the well-know granular search idea introduced by Toth and Vigo [60] to solve the CLRP and the MDVRP. Extensive computational experiments on benchmark instances for both problems have been performed to determine the effectiveness of the proposed algorithms.

This work is organized as follows:

- Chapter 1 describes a detailed overview and a methodological review of the literature for the the *Capacitated Location-Routing Problem* (CLRP) and the *Multi-Depot Vehicle Routing Problem* (MDVRP).

- Chapter 2 describes a two-phase hybrid heuristic algorithm to solve the CLRP.

- Chapter 3 shows a computational comparison of heuristic algorithms for the CLRP.

- Chapter 4 presents a hybrid granular tabu search approach for solving the MDVRP.

Keywords

Capacitated Location-Routing Problem

Multi-Depot Vehicle Routing Problem

Granular Tabu Search

Simulated Annealing

Variable Neighborhood Search

Heuristic Algorithms

Computational Comparison

Contents

Abstract **iii**

Keywords **iv**

1 Introduction **1**

1.1 The Capacitated Location-Routing Problem (CLRP) 1

1.1.1 Literature review for the CLRP 2

1.2 The Multi-Depot Vehicle Routing Problem (MDVRP) 4

1.2.1 Literature review for the MDVRP 5

2 Heuristic algorithm for the capacitated location-routing problem **7**

2.1 Description of the proposed algorithm 7

2.2 Procedure VRPH 8

2.3 Construction phase 9

2.3.1 Initial hybrid procedure 9

2.3.2 Splitting procedure 11

2.4 Improvement phase 12

2.4.1 Granular Neighborhoods 13

2.4.2 Space search and diversification strategies 14

2.5 Perturbation procedure 16

2.6 Computational results 17

2.6.1 Implementation details 17

2.6.2 Global results 20

2.6.2.1 Tuzun-Burke instances 21

2.6.2.2 Prodhon instances 21

2.6.2.3 Barreto instances 21

2.7 Concluding remarks 22

3 A comparison of heuristic algorithms for the CLRP 29

3.1 Introduction 29

3.2 General framework 30

3.2.1 Granular search space 30

3.2.2 Neighborhood structures 31

3.2.3 Initial solution 31

3.3 Description of the new proposed algorithms 31

3.3.1 The Granular Variable Tabu Neighborhood Search heuristic algorithm (GTVNS) 31

3.3.2 The Granular Simulated Annealing heuristic algorithm (GSA) 32

3.4 Computational experiments 33

3.4.1 Implementation details 34

3.4.2 Parameter settings 35

3.4.3 Comparison of the three described algorithms 36

3.4.4 Comparison of the effect of the initial solution 37

3.4.5 Comparison of the effect of the granularity 37

3.4.6 Global comparison 38

3.4.7 Comparison of the most efficient algorithms 38

3.5 Concluding remarks 40

4 A heuristic algorithm for the MDVRP 50

4.1 Hybrid Granular Tabu Search Algorithm 50

4.2 Initial Solution 51

4.3 Granular Tabu Search 54

4.3.1 Neighborhood Structures 54

4.3.2 Search, Intensification and Diversification strategies . . 56

4.3.3 Swapping Procedure 58

4.4 Computational experiments 58

4.4.1 Implementation details 58

4.4.2 Global results 61

4.5 Concluding remarks 62

Bibliography 65

List of Figures

2.1 Example of the splitting procedure 12
2.2 Example of Two-opt move by exchanging edges incident to the depots 14

List of Tables

2.1 Summarized results for each ingredient of 2-Phase HGTS on GAP BKS, GAP BKC and CPU time for the complete data set 23
2.2 Summarized results on GAP BKS, GAP BKC and CPU time for the complete data set 24
2.3 Summarized results on the number of BKS, BKC and new BKS for the
complete data set 25
2.4 Detailed results for the first data subset DS1 (Tuzun-Burke Instances) 26
2.5 Detailed results for the second data subset DS2 (Prodhon Instances) 27
2.6 Detailed results for the third data subset DS3 (Barreto Instances) 28

3.1 Summarized results on Gap PBKS by comparing
the quality of the Initial Solutions (G0 and B0) 41
3.2 Summarized results on Gap PBKS
without the "granular" search approach 42
3.3 Best results for 2-Phase HGTS, GTVNS,
and GSA on Tuzun-Burke Instances 43
3.4 Best results for 2-Phase HGTS, GTVNS,
and GSA on Prodhon Instances 44
3.5 Results for 2-Phase HGTS, GTVNS,
and GSA on Barreto Instances 45
3.6 Summarized best results for all the algorithms on the complete data set 46
3.7 Best results for all algorithms on Tuzun-Burke Instances . . . 47

3.8 Best results for all algorithms on Prodhon Instances 48
3.9 Best results for all algorithms on Barreto Instances 49

4.1 Solutions obtained by each phase of the proposed algorithm . 63
4.2 Solutions (CPU Times) obtained by the MDVRP Algorithms . 64

List of Algorithms

2.1 Procedure: VRPH 9

Chapter 1

Introduction

The *Location Routing Problem* (LRP) includes two types of fundamental problems of the supply chain management: the *Facility Location Problem* (FLP) and the *Vehicle Routing Problem* (VRP). The different aspects of these problems such as location, assignment and routing have been generally studied independently. This can be explained by considering that the location is a strategic decision which is taken for a long time frame, while the routing is an operational aspect which can be modified dynamically many times in a short time. However, it is well know that these decisions are interrelated. Indeed, the decision of locating a depot is often influenced by the transportation costs and vice versa (Rand [50]). As a consequence, the LRP has become an interesting field of research.

This work considers two problems: i) the LRP with capacity constrains for both the depots and the routes called the *Capacitated Location-Routing Problem* (CLRP), ii) the *Multi-Depot Vehicle Routing Problem* (MDVRP), which is a generalization of the well known *Vehicle Routing Problem* (VRP) by considering several depots.

1.1 The Capacitated Location-Routing Problem (CLRP)

The *Capacitated Location-Routing Problem* (CLRP) can be defined as follows: let $G = (V, E)$ be an undirected graph, where V is a set of nodes which is partitioned into a subset $I = 1, \ldots, m$ of potential depots and a subset

$J = 1, \dots, n$ of customers. Each potential depot $i \in I$ has a capacity w_i and an opening cost o_i. Each customer $j \in J$ has a nonnegative demand d_j which must be fulfilled by a depot. An unlimited set of identical vehicles, each with capacity q and fixed cost f , is available at each depot $i \in I$. Each edge $(i, j) \in E$ has an associated traveling cost c_{ij}. The goal of the CLRP is to determine the depots to be opened and the routes to be performed to fulfill the demand of the customers. Each route must start and finish at the same depot, the global demand of each route must not exceed the vehicle capacity q, and the global demand of the routes assigned to a depot $i \in I$ must not exceed its capacity w_i. The objective function of the CLRP is given by the sum of the costs of the open depots, of the costs of the traveled edges, and of the fixed costs associated with the used vehicles.

The *Capacitated Location-Routing Problem* (CLRP) is a strategic problem of the supply chain management. The basic hierarchical structure of the CLRP is a supply chain involving two echelons: depots and customers. The CLRP is an NP-hard problem, since it is a generalization of the two well known NP-hard problems: the *Capacitated Facility Location Problem* (CFLP) and the *Capacitated Vehicle Routing Problem* (CVRP). Indeed, the CFLP can be described as a CLRP with unlimited vehicle capacity (i.e. $q = \infty$), vehicle fixed cost equal to zero (i.e. $f = 0$), and infinite cost for the edges connecting any pair of customers (i.e. $c_{ij} = \infty$ for $i = m+1, \dots, m+n$ and $j = m+1, \dots, m+n$), and the CVRP can be described as a CLRP with only one depot (i.e. $m = 1$).

1.1.1 Literature review for the CLRP

Few surveys on location-routing problems have been presented in the literature. Min et al. [37] proposed a classification for the LRP based on the solution methods, and the problem perspectives. The most recent classification, proposed by Nagy and Salhi [40], is based on eight different aspects. This hierarchical taxonomy provides a more integrated view of the LRP literature. Different mathematical formulations with two and three indices have been proposed for the LRP and the CLRP. Three-index formulations for the LRP were introduced by Perl and Daskin [42] and Hansen et al. [26], and for the CLRP by Prins et al. [48]. Two-index formulations for the CLRP have been proposed by Laporte et al. [32], Baldacci et al. [4], Contardo et al.

[12], and Belenguer et al. [8]. These exact approaches can consistently solve to proven optimally small-medium size instances. For this reason, several heuristic algorithms have been proposed to solve large CLRP instances.

Nagy and Salhi [40] classified these algorithms as *sequential*, *iterative*, *hierarchical*, and *clustering based* methods. Sequential methods usually solve the facility location problem, and then the corresponding routing problem for each open depot (see, e.g. Daganzo [16]). According to Salhi and Rand [54], this type of approach avoids an important feedback between the two subproblems. On the other hand, iterative methods solve both subproblems in an iterative way providing a feedback between the two subproblems. In these methods, the CLRP is tackled either by solving the corresponding routing problem without considering the location decisions and assigning one depot for each cluster of customers, or by solving the facility location problem and performing at least one route for each open depot. Tuzun and Burke [61] proposed a two-phase tabu search approach that iterates between the location and the routing phases in order to search better solutions for large instances. In this work, results for instances with up to 200 customers have been reported. Prins et al. [48] proposed a two-phase algorithm which exchanges information between the location and routing phases. In the first phase, the routes and their customers are aggregated into super customers, and the corresponding CFLP is solved by using a Lagrangean relaxation technique. In the second phase, a granular tabu search (GTS) procedure (see Toth and Vigo [60]) with one neighborhood was used to solve the resulting MDVRP. At the end of each iteration, information about the promising edges is recorded to be used in the following phase.

Hierarchical methods solve the CLRP by using a hierarchical structure. First, the FLP is solved as the main problem, and then, the subsequent Routing Problem is solved as the subordinate problem. The location problem is solved in an approximate way by applying at each step a subroutine that solves the corresponding routing problem. Interested readers are referred to Albareda-Sambola et al. [2] and Melechovskỳ et al. [36].

Cluster based methods for the CLRP have been proposed by Barreto et al. [6]. In this work, in the first phase the customer set is split into clusters according to the vehicle capacity. In the second phase, a *Traveling Salesman Problem* (TSP) is solved for each cluster. Finally, in the final phase, the TSP circuits are grouped into super nodes for solving the corresponding CFLP.

Other heuristics for the CLRP have been proposed by Prins et al. [47]. In this work, a greedy randomized adaptive search procedure (GRASP), with a learning process and a path relinking strategy, has been proposed. A randomized version of the Clarke and Wright algorithm (proposed by Clarke and Wright [11] for the CVRP) is applied during the GRASP phase. In addition, a learning process is implemented to choose the correct depots. A path relinking strategy is then used as post optimization procedure to generate new solutions. The same authors (Prins et al. [46]) proposed a memetic algorithm with population management (MA|PM).

More recently, Duhamel et al. [18] developed a hybridized GRASP with an evolutionary local search (ELS) procedure. Yu et al. [66] proposed a Simulated Annealing (SA) heuristic based on three random neighborhoods. Pirkwieser and Raidl [43] proposed a Variable Neighborhood Search (VNS) coupled with ILP-based very large neighborhood searches to solve the (periodic) location-routing problem. An adaptive large neighborhood algorithm for the Two-Echelon Vehicle Routing Problem (2E-VRP), which is also able to solve the CLRP, has been introduced by Hemmelmayr et al. [29]. A GRASP with an ILP-based metaheuristic and a multiple ant colony optimization method have been proposed by Contardo et al. [13] and by Ting and Chen [59], respectively.

1.2 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 and E the edge set. The vertex set V is partitioned into a subset $I = 1, \dots, m$ of depots and a subset $J = 1, \dots, n$ 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.

1.2.1 Literature review for the MDVRP

The MDVRP is known to be a NP-hard, since it is a generalization of the well known Vehicle Routing Problem (VRP), arising when m = 1. Exact algorithms were proposed by Laporte et al. [31] and, recently, by Baldacci et al. [4]. Laporte et al. [33] 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 instances with less than 100 customers. 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 [64], Gillett and Johnson [22], Gillett and Miller [23], Golden et al. [24], and Raft [49]. All these methods use adaptations of VRP algorithms to solve the MDVRP. Chao et al. [9] 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. [24]. Finally, the current solution is improved by using a method based on a record-to-record approach proposed in Dueck [17]. Renaud et al. [52] 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. [51]. 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. [15] proposed a general tabu search heuristic which is also 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. [20]). 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 [44] proposed a unified heuristic, which is able to solve 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 [56], and the Ruin and Recreate paradigm introduced by Schrimpf et al. [55].

Evolutionary approaches for the MDVRP have been proposed by Thangiah and Salhi [58], Ombuki-Berman and Hanshar [41], and Vidal et al. [62]. Vidal et al. [62] proposed a metaheuristic based on the exploitation of a new population-diversity management mechanism to allow a broader access to re- production, 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 recent parallel iterated tabu search heuristic has been developed by Cordeau and Maischberger [14]. This heuristic combines tabu search with a simple perturbation procedure to allow the algorithm to explore new parts of the solution space.

Chapter 2

Heuristic algorithm for the capacitated location-routing problem

Notes about the chapter

The contents of this chapter is based on the paper entitled "*A two-phase hybrid heuristic algorithm for the capacitated location-routing problem*", co-authored with Rodrigo Linfati and Professor Paolo Toth, which has been published in Computers & Operations Research (ISSN: 0305-0548). Partial results were presented in the XVI CLAIO/SBPO, in Rio de Janeiro, Brazil (2012) and 5th International Workshop on Freight Transportation and Logistics (ODYSSEUS 2012), Mykonos – Greece.

2.1 Description of the proposed algorithm

This chapter presents a two-phase hybrid heuristic algorithm (2-Phase HGTS) developed for solving the CLRP. The main body of the proposed algorithm consists of two major phases: *Construction phase* and *Improvement phase*. In the *Construction phase*, the goal is to build an initial feasible solution using an *Initial hybrid procedure* followed by a *Splitting procedure* to minimize the routing cost. In the *Improvement phase,* a modified GTS procedure, which considers several diversification steps, is applied to improve the quality of the current solution. Whenever no improvement is obtained within $N_{pert} \times n$ iterations (where N_{pert} is a given parameter), the algorithm tries to escape from the current local optimum by applying a randomized *perturbation procedure.* In addition, a *procedure VRPH,* based on the library of local search heuristics for the VRP proposed by Groer et al. [25], is introduced as a general improvement routine.

The key-point for the success of the proposed algorithm is the location of the correct depots in the *Construction phase.* Since the most critical decisions of the *Improvement phase* are those concerning the opening and closing of the depots, a proper location of the depots is able to reduce the search space for the *Improvement phase* from a CLRP to a MDVRP. The previously mentioned procedures are described in more detail in the following subsections.

2.2 Procedure VRPH

Groer et al. [25] have recently proposed a software library containing fast local search heuristics for finding good feasible solutions for the CVRP. The standard library offers four different routines:

- *vrp_initial*: this routine uses a variant of the Clarke-Wright algorithm, proposed by Yellow [65], to generate initial solutions for the CVRP;
- *vrp_rtr*: this routine is an implementation of the record-to-record travel metaheuristic proposed by Li et al. [34];
- *vrp_sa*: this routine is an implementation of a Simulated Annealing (SA) metaheuristic;
- *vrp_ej*: this routine is an implementation of a neighborhood ejection/injection algorithm.

We developed a procedure, called VRPH, which applies routines *vrp_initial* and then, iteratively, routine *vrp_sa* and *vrp_rtr* until no improvement is reached. Procedure VRPH is executed in several parts of the two-phase hybrid algorithm as a general improvement procedure for a given depot. We do not use the ejection/injection algorithm *vrp_ej* since, according to our computational experiments on the considered CLRP benchmark instances, it increases a lot the global computing time with a negligible improvement of the quality solution. The outline of procedure VRPH is described in Algorithm 2.1.

Algorithm 2.1 Procedure: VRPH

```
input: vrp_instance, vrp_solution (optional)
output: vrp_solution

if no vrp_solution exists then
   vrp_initial(vrp_solution)
endif
repeat
   repeat
      call vrp_sa(vrp_solution)
   until vrp_solution is not improved
   repeat
      call vrp_rtr(vrp_solution)
   until vrp_solution is not improved
until vrp_solution is not improved
```

2.3 Construction phase

In this phase we propose a procedure to construct an initial feasible solution. The procedure is based on a hybrid methodology which combines exact and heuristic techniques. In addition, a cluster based method is considered as a starting point in an iterative framework. The *Construction phase* procedure calls in sequence the procedures *Initial hybrid* and *Splitting* described in the following subsections.

2.3.1 Initial hybrid procedure

The initial CLRP solution S_0 is obtained by applying a hybrid procedure which is generally able to find good feasible solutions within short computing times. This hybrid approach combines exact algorithms with the well-known Lin-Kernighan heuristic procedure (LKH) (see Lin and Kernighan [35] and Helsgaun [28]), used to find good solutions for the TSPs corresponding to the routes defined by a depot and a subset of customers.

A good initial CLRP solution can be obtained by recognizing clusters of customers which can be visited in the same route. To this end, we have developed a procedure that considers all the customers and constructs the corresponding giant TSP tour by using procedure LKH. The giant tour is then split into several clusters so as to satisfy for each cluster the vehicle capacity. Then, for each depot i and for each cluster j, procedure LKH is applied to find the corresponding TSP tour, and to get the route cost l_{ij} for assigning depot i to cluster j. The best assignment of the depots to the clusters is obtained by introducing two sets of binary variables x and y, where $x_{ij} = 1$ iff depot i is assigned to cluster j, and $y_i = 1$ iff depot i is opened, and by solving the following integer linear programming (ILP) model:

$$min\, z = \sum_{i \in D} O_i y_i + \sum_{i \in D} \sum_{j \in G} l_{ij} x_{ij} \tag{2.1}$$

subject to

$$\sum_{i \in D} x_{ij} = 1 \qquad \forall j \in G \tag{2.2}$$

$$\sum_{j \in G} dc_j x_{ij} \leqslant W_i y_i \qquad \forall i \in D \tag{2.3}$$

$$y_i \in \{0, 1\} \qquad \forall i \in D \tag{2.4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in D, j \in G \tag{2.5}$$

where:
D *set of depots*
G *set of clusters*
dc_j *global demand of cluster j*

The objective function (2.1) sums the opening costs for of the used depots and the traveling costs associated with the edges traversed by the routes. Constraints (2.2) guarantee that each cluster is assigned to exactly one depot. Constraints (2.3) impose the capacity for the open depots. Finally, constraints (2.4) and (2.5) impose the integrality of the variables used in the model. It has to be noted that ILP model (2.1)-(2.5) corresponds to the formulation of the well known *Single Source Capacitated Plant Location*

Problem (see, e.g. Barcelo and Casanovas [5], and Klincewicz and Luss [30]).

It is worth to that there are n possibilities to split the giant tour, by considering each customer as possible initial vertex. For this reason, the hybrid procedure is repeated n times keeping the best feasible solution found. The proposed algorithm tries to improve the current solution by applying the *Splitting procedure* described in the following subsection.

2.3.2 Splitting procedure

The *Splitting procedure* is based on the idea that the total traveling cost can be decreased by adding new routes, and assigning them to different depots. Note that the splitting procedure can be effective only when the cost F for using a vehicle is small. The procedure starts by considering the route which contains the longest (largest cost) edge and by selecting its three longest edges. Then, for the three combinations of two of these edges, say edges (r, s) and (t, u), the following steps are performed (see Fig. 2.1):

- edges (r, s) and (t, u) are removed from the considered route;
- the considered 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 open depot for which the assignment of the cluster satisfies the depot capacity constraint, procedure LKH is applied to find the TSP tour corresponding to the assignment of the cluster to the depot;
- the cluster is assigned to the depot, say d, for which the cost of the corresponding TSP tour is minimum;
- procedure VRPH is applied to the customers currently assigned to depot d, and to those currently assigned to the depot associated with the considered route (for both depots, the associated current CVRP solutions are given on input to procedure VRPH).

Whenever the global cost of the new solution is smaller than that of the best solution found so far, the latter solution is updated. We repeat

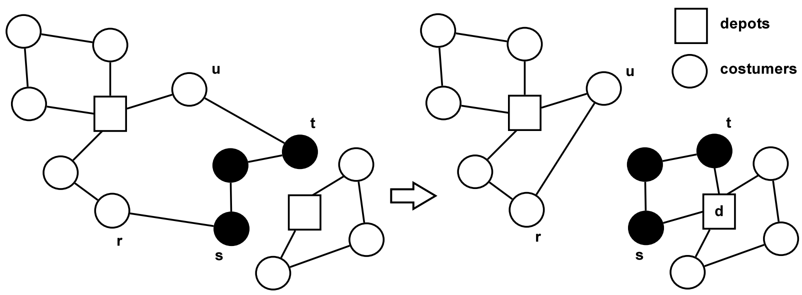

Figure 2.1: Example of the splitting procedure

the *Splitting procedure* N_{split} times (where N_{split} 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 *Initial hybrid procedure* has not been changed.

2.4 Improvement phase

In this stage, the algorithm tries to improve the initial solution S_0 obtained by the *Construction phase* applying a modified granular tabu search (GTS) procedure. The goal of the Improvement phase is to optimize the routes without considering moves between close and open depots, hence the search space is related to a MDVRP. In this phase, we allow infeasible solutions with respect to the depot and vehicle capacities (see subsection 3.3.2).

To reduce the computing time required by each iteration of a local search procedure, which can steeply grow with the instance size, Toth and Vigo [60] proposed the so called *granular tabu search* (GTS) approach. The method is based on the use of a candidate list strategy, which drastically reduces the time required by a tabu search algorithm. The main objective of the GTS approach is to have good solutions by using a neighborhood structure that can be evaluated in a short time. Three main differences with respect to the idea of "granularity" introduced by Toth and Vigo [60] for the CVRP are considered here. Basically, the proposed algorithm considers five neighborhoods, three different diversification strategies, and a random perturbation procedure to avoid that the algorithm remains in a local optimum for a given number of iterations.

If the number of routes of the current solution is greater than the minimum number of routes, N_{min}, required to visit all the customers, where $N_{min} = \left\lceil \frac{\sum_{j=m+1}^{m+n} d_j}{Q} \right\rceil$, an attempt is performed to reduce the number of routes. In particular, the algorithm 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 $F_2(S)$ described in subsection 3.3.2, of one of the remaining routes. A new solution S is then determined by applying procedure VRPH for all the depots involved in the move for which the depot capacity constraint is satisfied. For each depot, the corresponding CVRP solutions are given on input to procedure VRPH. The proposed granular neighborhoods, diversification strategies and perturbation procedure are described in the following subsections.

2.4.1 Granular Neighborhoods

The proposed algorithm executes the following five types of moves for *Max_Iter* iterations (where *Max_Iter* is a given parameter):

- Shift: One customer is transferred from its current position to another position either in the same or in a different route (assigned to the same or to a different depot).

- Swap: Two customers are exchanged, either in the same route or between different routes (assigned to the same or to different depots).

- Two opt: This is a modified version of the well-known 2-opt move, in which two non consecutive edges are removed and the routes are reconnected in a different way. Note that if the two selected edges are in the same route, the two opt move is equivalent to that described by Lin and Kernighan [35]. If the two edges are in different routes assigned to the same depot, the move is similar to the traditional 2-opt inter route move for the VRP. Otherwise, if the edges belong to different depots, there are several ways to rearrange the routes. In this case, it is necessary to perform an additional move concerning the edges connecting the depots with the last customers of the selected routes to ensure that each route starts and finishes at the same depot.

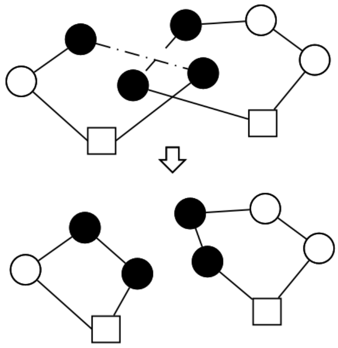

Figure 2.2: Example of Two-opt move by exchanging edges incident to the depots

- Exchange: Two consecutive customers are transferred from their current positions to different positions by keeping the edge connecting them. The two customers can be inserted in their current route or in a different route (assigned to the same or to a different depot).

- Inter-tour exchange: This is an extension of the Swap move and considers two pairs of consecutive customers. The edge connecting each pair of customers is kept. The exchange is performed between two different routes (assigned to the same or to different depots).

2.4.2 Space search and diversification strategies

The proposed GTS procedure uses the same space search introduced by Toth and Vigo [60]. The original complete graph G is replaced by a sparse graph which includes all the edges whose cost is smaller than the *granularity threshold* ϑ, the edges incident to the depot, and those belonging to the best solution found so far. The value of ϑ is defined by means of an increasing function of the *sparsification factor* β: $\vartheta = \beta \bar{z^*}$, where $\bar{z^*}$ is the average cost of the edges in the current best solution found so far. Only the moves for which all the involved edges are contained in the sparse graph are considered.

Three diversification strategies have been considered. The first strategy is related to the granularity diversification proposed by Toth and Vigo [60]. Initially, the sparsification factor β is set to its initial value β_0. If no improvement of the best feasible solution found so far is reached after $N_{movbeta}$ iterations, the sparsification factor β is increased to β_d. A new sparse graph is then calculated, and $N_{moviter}$ iterations are executed starting from the best solution found so far. Finally, the sparsification factor β is reset to its initial value β_0 and the search continues. β_0, β_d, $N_{movbeta}$ and $N_{moviter}$ are given parameters.

The second diversification strategy is based on a *penalty approach*. Since infeasible solutions can be considered during the search process, we have implemented the following penalty scheme based on the techniques proposed by Gendreau et al. [21] and Taillard [57] for the VRP. Let us consider a CLRP solution S composed by a set of k routes $R_1, \ldots, R_k$. Each route R_r, $r \in \{1, \ldots, k\}$, is denoted by $(v_{r0}, v_{r1,} v_{r2}, \ldots, v_{r0})$, where v_{r0} represents the open depot assigned to the route, and v_{r1}, $v_{r2}, \ldots$ represent the visited customers. Note that S can be feasible or infeasible with respect to the vehicle capacity and the depot capacity. Let T be the subset of the open depots. In addition, the following notation is used: $v \in R_r$ if a customer v belongs to route R_r, $(u, v) \in R_r$ if u and v are two consecutive vertices of route R_r, and D_i is the set of customers assigned to the open depot i. The following objective function $F_1(S)$ is associated with any feasible solution S:

$$F_1(S) = \sum_{i \in T} O_i + \sum_{r=1}^{k} \sum_{(u,v) \in R_r} c_{uv} + Fk$$

The following objective function $F_2(S)$ is associated with any solution S (feasible or infeasible):

$$F_2(S) \;\; = \;\; F_1(S) + P_d \sum_{i \in T} \left[\sum_{v \in D_i} d_v - W_i \right]^+ + P_r \sum_{r=1}^{k} \left[\sum_{v \in R_r} d_v - Q \right]^+$$

where $[x]^+ = max(0, x)$, and P_d and P_r are two positive weights used to increase the cost of the solution S by adding the sum of the excess loads of the overloaded open depots, and the sum of the excess demands of the overloaded routes, respectively. The two weights are calculated as follow: $P_d = \alpha_d \times F_1(S_0)$ and $P_r = \alpha_r \times F_1(S_0)$, where $F_1(S_0)$ is the value of the objective function of the solution S_0 obtained by the *Construction phase*, and α_d and α_r are two parameters which are adjusted during the search within the range $[\alpha_{min}, \alpha_{max}]$. In particular, if no infeasible solutions with respect to the depot capacity have been found over N_{movpen} iterations, then the value of α_d is set to $max\{\alpha_{min}, \alpha_d \times r_{pen}\}$, where $r_{pen} < 1$. On the other hand, if no feasible solutions have been found during N_{movpen} iterations, then the value of α_d is set to $min\{\alpha_{max}, \alpha_d \times i_{pen}\}$, where $i_{pen} > 1$. A similar rule is applied to modify the value of α_r. α_d, α_r, α_{min}, α_{max}, N_{movpen}, r_{pen}, i_{pen} are given parameters.

In the selection of the best move to be performed we consider the following criterion for the evaluation of a move leading to an infeasible solution S. If the value of $F_2(S)$ is less than the cost of the best solution found so far, we assign S a value $F(S) = F_2(S)$. Otherwise, as diversification strategy, we introduce an extra penalty by adding to $F_2(S)$ a constant term equal to the product of the absolute difference value $\triangle_{max}$ between two successive values of the objective function, the square root of the number of routes k, and a scaling factor g (for further details see Taillard [57]). Therefore, we define $F(S) = F_2(S) + \triangle_{max}\sqrt{k}g$ (where g is a given parameter). Note that if the new solution S is feasible, we define $F(S) = F_1(S)$. The move corresponding to the minimum value of $F(S)$ is performed.

In the third diversification strategy, every $N_{fact} \times n$ iterations (where N_{fact} is a given parameter), we consider the best solution found so far which is feasible with respect to the depot capacity and apply procedure VRPH for each open depot. Note that procedure VRPH is able to transform a solution which is infeasible with respect to the route capacity into a feasible solution. This diversification strategy may help the algorithm to explore new parts of the solution space.

2.5 Perturbation procedure

Since the modified GTS procedure can fail in finding a move improving the current solution, the algorithm tries to escape from a local optimum by perturbing the current solution. In particular, if no improving move has been performed after $N_{pert} \times n$ iterations, the algorithm applies a perturbation approach similar to the "3-route procedure" proposed by Renaud et al. [52].

Differently from what is proposed by Renaud et al. [52], we consider a randomized procedure for selecting the routes to be perturbed. In particular, we use an exchange scheme involving three routes. The algorithm selects the first route $k1$ in a random way. The second route $k2$ is the closest neighbor of $k1$, and the third route $k3$ is the closest neighbor of $k2$, with $k1 \neq k3$. The evaluation of the "distance" between the routes depends on the characteristics of the considered instance. In particular, as it is the case for the benchmark instances considered in our computational experiments (see Section 4), if each vertex of the input graph G is associated with a point in the plane, and the cost c_{ij} of edge (i, j) in proportion to the Euclidean distance between the points associated with vertices i and j, then the distance between the routes is calculated by considering their "center of gravity".

For each customer $i1$ of route $k1$, each customer $i2$ of route $k2$, each edge $(h2, j2)$ of route $k2$ (with $h2 \neq i2$ and $j2 \neq i2$), and each edge $(h3, j3)$ of route $k3$, we obtain a new solution S by considering the following move, in which we do not impose the depot and vehicle capacity constraints:

- remove customer $i1$ from route $k1$ and insert it between vertices $h2$ and $j2$ in route $k2$;
- remove customer $i2$ from route $k2$ and insert it between vertices $h3$ and $j3$ in route $k3$.

The move associated with the solution S corresponding to the minimum value of $F_2(S)$ is performed, even if S is worse than the current solution.

2.6 Computational results

2.6.1 Implementation details

The overall algorithm (2-Phase HGTS) has been implemented in C++, and the computational experiments have been performed on an Intel Core Duo CPU (2.00 GHz) under Linux Ubuntu 11.04 with 2 GB of memory. The ILP model (2.1) - (2.5) has been optimally solved by using the ILP solver CPLEX 12.1. The performance of the proposed algorithm has been evaluated by considering 79 benchmark instances taken from the literature. The complete set of instances considers three data subsets. The first data subset (DS1) was proposed by Tuzun and Burke [61] and considers 36 instances with capacity constraints only on the routes. It considers instances with $n = 100$, 150 and 200 customers. The number m of potential depots is either 10 or 20. The customers and the depots correspond to random points in the plane. The traveling cost of an arc is calculated as the Euclidean distance between the points corresponding to the extreme vertices of the arc. The vehicle capacity Q is set to 150, and the demands of the customers are uniformly random distributed in the interval $[1, 20]$.

The second data subset (DS2) was proposed by Prins et al. [45], and contains 30 instances with capacity constraints on both the routes and the depots. The number m of potential depots is either 5 or 10, and the number of customers is $n = 20$, 50, 100 and 200. The customers and the depots correspond to random points in the plane. For this data subset, the traveling costs are calculated as the corresponding Euclidean distances, multiplied by 100 and rounded up to the next integer. The vehicle capacity Q is either 70 or 150, and the demands of the customers are uniformly random distributed in the interval$[11, 20]$.

The instances of the third data subset (DS3), introduced by Barreto [7], were obtained from some classical CVRP instances by adding new depots with the corresponding capacities and fixed costs. This data subset considers 13 instances. The routes are capacitated and, with the exception of few instances, the depots are also capacitated. The number of customers ranges from 21 to 150, and the number of potential depots from 5 to 10.

For each instance, only one run of the proposed algorithm is executed. The total number of iterations of the main loop on the *Improvement Phase*, *Max_Iter*, is set to $10 \times n$. The tabu tenure for each move performed is calculated (as in Gendreau et al. [21]) as an integer uniformly distributed random number in the interval $[5, 10]$. As for other heuristics, extensive computational tests have been made to find a suitable set of parameters. On average, the best performance of 2-Phase HGTS has been obtained by considering the following values of the parameters: $N_{pert} = 0.20$, $N_{split} = 7$, $\beta_0 = 1.50$, $\beta_d = 2.40$, $N_{movbeta} = 2$, $N_{moviter} = 1$, $\alpha_d = 0.01$, $\alpha_r = 0.0075$, $\alpha_{min} = \frac{1}{F_1(S_0)}$, $\alpha_{max} = 0.04$, $N_{movpen} = 10$, $i_{pen} = 2.00$, $r_{pen} = 0.30$, $g = 0.02$, and $N_{fact} = 1.50$. These values have been utilized for the solution of all the considered instances.

The proposed algorithm has been compared (see Tables 2.2 to 2.6) with the five most effective published heuristics proposed for the CLRP: GRASP of Prins et al. [47], the memetic algorithm with population management (MA|PM) of Prins et al. [46], the Langrangean relaxation and granular tabu search method (LRGTS) of Prins et al. [48], GRASP+ELS of Duhamel et al. [18], and the simulated annealing algorithm (SALRP) of Yu et al. [66]. The results reported for GRASP (Prins et al. [47]), MA|PM (Prins et al. [46]), LGRTS (Prins et al. [48]) and SALRP (Yu et al. [66]) correspond to a single run of the associated algorithm. GRASP+ELS (Duhamel et al. [18]) has been run five times by considering five different random generator seeds, and the reported cost is the best found over the five runs; the reported computing time is the time required to reach the best solution within the corresponding run. In the paper by Yu et al. [66], the authors report also the cost of the best solution found by SALRP during the parameter analysis phase. In Tables 2.1 to 2.6, the following notation is used:

Instance	instance name;
n	number of customers;
m	number of potential depots;
Cost	solution cost obtained by each algorithm (either one single run or the best run);
BKC	cost of the best-known result among GRASP, MA|PM, LRGTS, GRASP+ELS, SALRP and 2-Phase HGTS;
BKS	cost of the best-known result obtained either by the six considered algorithms (BKC) or during the parameter analysis phase of SALRP;
CPU	CPU used by each method;
CPU index	Passmark performance Test for each CPU;
CPU time	running time in seconds on the CPU used by each algorithm;
Gap BKC	percentage gap of the solution cost found by each algorithm with respect to BKC;
Gap BKS	percentage gap of the solution cost found by each algorithm with respect to BKS.

In addition, for each instance, the costs which are equal to the corresponding BKC, are reported in bold. Whenever algorithm 2-Phase HGTS improves the BKS value, its result is underlined. Finally, the CPU index is given by the Passmark performance test[1]. This is a well known bench-

[1]PassMark® Software Pty Ltd, http://www.passmark.com

mark test focused on CPU and memory performance. Higher values of the Passmark test indicate that the corresponding CPU is faster.

2.6.2 Global results

Table 2.1 provides the contribution of each of the ingredients of the proposed heuristic to the quality of the final solution. The table shows the results (average values of Gap BKS, Gap BKC and the cumulative CPU time) corresponding to each of the following solutions:

- Initial hybrid: solutions obtained after the application of the Initial hybrid procedure;

- Splitting: solutions obtained after the application of the Splitting procedure (i.e. at the end of the First Phase);

- Global: solutions obtained by the proposed 2-Phase HGTS heuristic (i.e. at the end of the Second Phase).

In addition, the results corresponding to the solutions obtained at the end of the Second Phase "without" a specific ingredient, but with all the other ingredients active have been reported. The following solutions have been considered:

- Wsecond: solutions obtained without considering the second diversification strategy;

- Wthird: solutions obtained without considering the third diversification strategy;

- Wperturbation: solutions obtained without considering the perturbation procedure.

The Splitting procedure is rather time consuming, but it produces substantial improvements on all the instances. The table shows that each of the ingredients used in the proposed algorithm is effective.

A summary about the results obtained by the considered six algorithms for the complete instance dataset is given in Tables 2.2 and 2.3. Table 2.2 provides the average values of Gap BKS, Gap BKC and CPU time, and the CPU index of the corresponding CPU. Table 2.3 reports the number of BKC, BKS and new best known (new BKS) solutions obtained by each algorithm. Table 2.2 shows that the proposed algorithm provides the lowest global averages for Gap BKS and Gap BKC. As for the global CPU time, the proposed algorithm is faster than GRASP+ELS and SALRP, which were able to find the previous best results in terms of average gaps and number of best solutions. It is to note that the CPU time reported for algorithm GRASP+ELS does not represent the global time required to find the best solution (obtained by executing five runs), since it corresponds to the CPU time spent, for each instance, in a single run. On the other hand, the CPU time of 2-Phase HGTS is larger than that of GRASP, MA|PM and LGRTS. This can be explained by the fact that we use several improvement procedures in the second phase. Although the CPU time of the proposed algorithm is larger than that of these approaches, it remains within an acceptable range for a strategic problem like CLRP. In addition, algorithm 2-Phase HGTS is able to find the largest number of best solutions.

2.6.2.1 Tuzun-Burke instances

The results for the first data subset (DS1) are shown in Table 2.4. The results show that the proposed algorithm outperforms all the other heuristics for what concerns the global average values of Gap BKS and Gap BKC, and the global number of the best solutions found. It is to note that the performance of the proposed algorithm improves, with respect to that of the other methods, for the largest instances (150 and 200 customers).

2.6.2.2 Prodhon instances

The detailed results for the second data subset (DS2) are given in Table 2.5. On average, the proposed approach has values of Gap BKS and Gap BKC smaller than those of GRASP, MA|PM, LRGTS, and GRASP+ELS. Only SALRP provides, although with longer CPU times, slightly better values of Gap BKS and Gap BKC. It is worth to note that the proposed algorithm clearly outperforms all the other methods for large-scaled instances with 200 customers.

2.6.2.3 Barreto instances

The results obtained by the proposed algorithm and by the other approaches for the third data subset (DS3) are given in Table 2.6. The table shows that the proposed algorithm is competitive with the other algorithms in terms of solution quality.

2.7 Concluding remarks

We propose an effective two-phase hybrid heuristic algorithm for the capacitated location routing problem (CLRP). In the proposed heuristic, after the construction of an initial feasible solution in the *Construction phase*, we apply an *Improvement phase* based on a modified Granular Tabu Search which considers five granular neighborhoods, three different diversification strategies and a perturbation procedure. The perturbation procedure is applied whenever the algorithm remains in a local optimum for a given number of iterations.

We compared the proposed algorithm with the five most effective published heuristics for the CLRP on a set of benchmark instances from the literature. The results show the effectiveness of the proposed algorithm, and several best known solutions are improved within reasonable computing times. The results obtained suggest that the proposed framework could be applied to other problems as the periodic location-routing problem (PLRP), the multi depot vehicle routing problem (MDVRP) and several extensions of the CLRP obtained by adding constraints as time windows, heterogeneous fleet, etc.

Table 2.1: Summarized results for each ingredient of 2-Phase HGTS on GAP BKS, GAP BKC and CPU time for the complete data set

Set	Size	Initial hybrid			Splitting			Global			Wsecond			Wthird			Wperturbation		
		Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time
DS1	36	7.05	6.87	91	1.23	1.05	298	0.68	0.51	392	0.80	0.62	376	0.90	0.73	367	0.79	0.62	376
DS2	30	3.41	3.29	46	1.28	1.17	117	0.49	0.38	176	0.82	0.70	166	0.99	0.87	150	0.85	0.74	166
DS3	13	6.86	6.80	12	1.74	1.69	87	0.78	0.74	105	0.97	0.92	100	0.97	0.92	98	0.78	0.74	101
Global Avg.		**5.64**	**5.50**	**61**	**1.33**	**1.20**	**195**	**0.63**	**0.50**	**263**	**0.84**	**0.70**	**251**	**0.95**	**0.81**	**240**	**0.81**	**0.69**	**251**

Table 2.2: Summarized results on GAP BKS, GAP BKC and CPU timefor the complete data set

Set	Size	GRASP			MA|PM			LRGTS			GRASP + ELS			SALRP			2-Phase HGTS		
		Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time	Gap BKS	Gap BKC	CPU time
DS1	36	3.03	2.85	163	1.40	1.23	207	1.38	1.20	22	0.83	0.66	607	1.03	0.85	826	**0.68**	**0.51**	392
DS2	30	3.57	3.45	97	1.35	1.23	96	0.71	0.59	18	1.04	0.92	258	**0.38**	**0.27**	422	0.49	0.38	176
DS3	13	1.63	1.58	20	2.06	2.01	36	1.66	1.61	18	**0.08**	**0.03**	188	0.29	0.25	161	0.78	0.74	105
Global Avg.		3.00	2.87	114	1.49	1.36	137	1.17	1.04	20	0.79	0.65	405	0.66	0.53	564	**0.63**	**0.50**	263
CPU		Intel Pentium 4 (2.40 Ghz)			Intel Pentium 4 (2.40 Ghz)			Intel Pentium 4 (2.40 Ghz)			Intel Core2 Quad (2.83 Ghz)			Intel Core2 Quad (2.66 Ghz)			Intel Core2 Duo (2.00 Ghz)		
CPU index		314			314			314			4373			4046			1398		

Table 2.3: Summarized results on the number of BKS, BKC and new BKS for thecomplete data set

	GRASP	MA|PM	LRGTS	GRASP+ELS	SALRP	2-Phase HGTS
DS1 (36 Instances)						
Total BKC	0	5	0	14	7	**18**
Total BKS	0	0	0	6	5	**14**
New BKS	0	0	0	2	1	**7**
DS2 (30 Instances)						
Total BKC	4	11	6	13	**15**	14
Total BKS	4	10	5	**12**	11	9
New BKS	0	0	0	2	2	**3**
DS2 (13 Instances)						
Total BKC	4	5	2	**11**	**11**	8
Total BKS	4	5	2	**10**	**10**	7
New BKS	0	0	0	**1**	**1**	0
BKC overall	8	21	8	38	33	**40**
BKS overall	8	15	7	28	26	**30**
New BKS overall	0	0	0	5	4	**10**

Table 2.4: Detailed results for the first data subset DS1 (Tuzun-Burke Instances)

Instance	n	m	BKS	BKC	GRASP Cost	GRASP Gap BKS	GRASP Gap BKC	GRASP CPU time	MA|PM Cost	MA|PM Gap BKS	MA|PM Gap BKC	MA|PM CPU time	LRGTS Cost	LRGTS Gap BKS	LRGTS Gap BKC	LRGTS CPU time	GRASP + ELS Cost	GRASP + ELS Gap BKS	GRASP + ELS Gap BKC	GRASP + ELS CPU time	SALRP Cost	SALRP Gap BKS	SALRP Gap BKC	SALRP CPU time	2-Phase HGTS Cost	2-Phase HGTS Gap BKS	2-Phase HGTS Gap BKC	2-Phase HGTS CPU time
111112	100	10	1467.68	1473.36	1525.25	3.92	3.52	33	1493.92	1.79	1.40	33	1490.82	1.58	1.19	3	**1473.36**	0.39	0.00	233	1477.24	0.65	0.26	369	1479.21	0.79	0.40	152
111122	100	20	1449.20	1449.20	1526.90	5.36	5.36	41	1471.36	1.53	1.53	36	1471.76	1.56	1.56	8	**1449.20**	0.00	0.00	9	1470.96	1.50	1.50	274	1486.27	2.56	2.56	239
111212	100	10	1394.80	1396.59	1423.54	2.06	1.93	28	1418.83	1.72	1.59	36	1412.04	1.24	1.11	4	**1396.59**	0.13	0.00	112	1408.65	0.99	0.86	231	1407.26	0.89	0.76	120
111222	100	20	1432.29	1432.29	1482.29	3.49	3.49	36	1492.46	4.20	4.20	36	1443.06	0.75	0.75	8	**1432.29**	0.00	0.00	114	**1432.29**	0.00	0.00	420	1474.01	2.91	2.91	146
112112	100	10	1167.16	1167.16	1200.24	2.83	2.83	28	1173.22	0.52	0.52	33	1187.63	1.75	1.75	8	**1167.16**	0.00	0.00	27	1177.14	0.86	0.86	348	**1167.16**	0.00	0.00	232
112122	100	20	1102.24	1102.24	1123.64	1.94	1.94	34	1115.37	1.19	1.19	43	1115.95	1.24	1.24	8	**1102.24**	0.00	0.00	259	1110.36	0.74	0.74	342	**1102.24**	0.00	0.00	224
112212	100	10	791.66	791.66	814.00	2.82	2.82	23	793.97	0.29	0.29	38	813.28	2.73	2.73	5	792.03	0.05	0.05	5	**791.66**	0.00	0.00	360	**791.66**	0.00	0.00	201
112222	100	20	728.30	728.30	747.84	2.68	2.68	38	730.51	0.30	0.30	49	742.96	2.01	2.01	6	**728.30**	0.00	0.00	48	731.95	0.50	0.50	418	**728.30**	0.00	0.00	254
113112	100	10	1238.49	1238.49	1273.10	2.79	2.79	23	1262.32	1.92	1.92	38	1267.93	2.38	2.38	4	1240.39	0.15	0.15	55	**1238.49**	0.00	0.00	300	**1238.49**	0.00	0.00	160
113122	100	20	1245.31	1246.00	1272.94	2.22	2.16	36	1251.32	0.48	0.43	48	1256.12	0.87	0.81	6	**1246.00**	0.06	0.00	233	1247.28	0.16	0.10	428	1251.22	0.47	0.42	237
113212	100	10	902.26	902.26	912.19	1.10	1.10	20	903.82	0.17	0.17	35	913.06	1.20	1.20	4	902.30	0.00	0.00	249	**902.26**	0.00	0.00	291	**902.26**	0.00	0.00	135
113222	100	20	1018.29	1018.29	1022.51	0.41	0.41	38	1022.93	0.46	0.46	63	1025.51	0.71	0.71	5	**1018.29**	0.00	0.00	196	1024.02	0.56	0.56	316	**1018.29**	0.00	0.00	157
Avg.						**2.64**	**2.59**	**32**		**1.22**	**1.17**	**41**		**1.50**	**1.45**	**6**		**0.06**	**0.02**	**128**		**0.50**	**0.45**	**341**		**0.64**	**0.59**	**188**
131112	150	10	1922.59	1944.57	2006.70	4.37	3.20	113	1959.39	1.91	0.76	129	1946.01	1.22	0.07	13	**1944.57**	1.14	0.00	518	1953.85	1.63	0.48	743	1961.75	2.04	0.88	485
131122	150	20	1833.95	1856.51	1888.90	3.00	1.74	161	1881.67	2.60	1.36	144	1875.79	2.28	1.04	19	1864.24	1.65	0.42	705	1899.05	3.55	2.29	835	**1856.51**	1.23	0.00	298
131212	150	10	1978.27	1984.25	2033.93	2.81	2.50	100	**1984.25**	0.30	0.00	111	2010.53	1.63	1.32	11	1992.41	0.71	0.41	727	2057.53	4.01	3.69	456	2012.69	1.74	1.43	406
131222	150	20	1801.39	1801.39	1856.07	3.04	3.04	133	1855.25	2.99	2.99	144	1819.89	1.03	1.03	16	1835.25	1.88	1.88	415	**1801.39**	0.00	0.00	833	1803.01	0.09	0.09	302
132112	150	10	1445.25	1445.25	1508.33	4.36	4.36	118	1448.27	0.21	0.21	168	1448.65	0.24	0.24	23	1453.78	0.59	0.59	103	1453.30	0.56	0.56	750	**1445.25**	0.00	0.00	449
132122	150	20	1441.98	1444.17	1456.82	1.03	0.88	166	1459.83	1.24	1.08	155	1492.86	3.53	3.37	28	**1444.17**	0.15	0.00	662	1455.50	0.94	0.78	828	1452.07	0.70	0.55	493
132212	150	10	1204.42	1204.42	1240.40	2.99	2.99	134	1207.41	0.25	0.25	201	1211.07	0.55	0.55	19	1219.86	1.28	1.28	459	1206.24	0.15	0.15	752	**1204.42**	0.00	0.00	270
132222	150	20	930.99	931.49	940.80	1.05	1.00	143	934.79	0.41	0.35	196	936.93	0.64	0.58	14	945.81	1.59	1.54	224	934.62	0.39	0.34	842	**931.49**	0.05	0.00	335
133112	150	10	1704.58	1705.36	1736.90	1.90	1.85	93	1720.30	0.92	0.88	144	1729.31	1.45	1.40	18	1712.11	0.44	0.40	271	1720.81	0.95	0.91	742	**1705.36**	0.05	0.00	444
133122	150	20	1400.01	1402.94	1425.74	1.84	1.63	128	1429.34	2.09	1.88	156	1424.59	1.76	1.54	19	**1402.94**	0.21	0.00	524	1415.85	1.13	0.92	833	1416.74	1.19	0.98	342
133212	150	10	1201.23	1203.44	1223.70	1.87	1.68	89	**1203.44**	0.18	0.00	154	1216.32	1.26	1.07	15	1214.82	1.13	0.95	251	1216.84	1.30	1.11	756	1234.83	2.80	2.61	526
133222	150	20	1152.18	1155.96	1231.33	6.87	6.52	135	1158.54	0.55	0.22	223	1162.16	0.87	0.54	14	**1155.96**	0.33	0.00	375	1159.12	0.60	0.27	837	1156.05	0.34	0.01	380
Avg.						**2.93**	**2.62**	**126**		**1.14**	**0.83**	**160**		**1.37**	**1.06**	**17**		**0.93**	**0.62**	**436**		**1.27**	**0.96**	**767**		**0.85**	**0.55**	**394**
121112	200	10	2265.59	2265.59	2384.01	5.23	5.23	385	2293.99	1.25	1.25	523	2296.52	1.37	1.37	41	2295.90	1.34	1.34	655	2324.10	2.58	2.58	1328	**2265.59**	0.00	0.00	522
121122	200	20	2166.43	2166.43	2288.09	5.62	5.62	410	2277.39	5.12	5.12	458	2207.50	1.90	1.90	40	2203.57	1.71	1.71	432	2258.16	4.23	4.23	1455	**2166.43**	0.00	0.00	603
121212	200	10	2245.33	2246.39	2273.19	1.24	1.19	311	2274.57	1.30	1.25	378	2260.87	0.69	0.64	33	**2246.39**	0.05	0.00	1566	2260.30	0.67	0.62	1319	2249.40	0.18	0.13	527
121222	200	20	2237.81	2237.81	2345.10	4.79	4.79	419	2376.25	6.19	6.19	436	2259.52	0.97	0.97	40	2265.53	1.24	1.24	2192	2326.53	3.96	3.96	1428	**2237.81**	0.00	0.00	558
122112	200	10	2089.77	2106.26	2137.08	2.26	1.46	338	**2106.26**	0.79	0.00	351	2120.76	1.48	0.69	48	2106.47	0.80	0.01	1521	2112.65	1.09	0.30	1320	2121.93	1.54	0.74	522
122122	200	20	1719.96	1722.99	1807.29	5.08	4.89	370	1771.53	3.00	2.82	378	1737.81	1.04	0.86	59	1779.05	3.44	3.25	618	**1722.99**	0.18	0.00	1400	1749.10	1.69	1.52	691
122212	200	10	1466.62	1467.54	1496.75	2.05	1.99	243	**1467.54**	0.06	0.00	323	1488.55	1.50	1.43	38	1474.25	0.52	0.46	514	1469.10	0.17	0.11	1299	1473.27	0.45	0.39	724
122222	200	20	1082.59	1082.59	1095.92	1.23	1.23	309	1088.00	0.50	0.50	505	1090.59	0.74	0.74	39	1085.69	0.29	0.29	1243	1088.64	0.56	0.56	1429	**1082.59**	0.00	0.00	616
123112	200	10	1970.44	1973.28	2044.66	3.77	3.62	283	**1973.28**	0.14	0.00	413	1984.06	0.69	0.55	43	2004.33	1.72	1.57	1451	1994.16	1.20	1.06	1318	1984.77	0.73	0.58	542
123122	200	20	1918.93	1932.05	2090.95	8.96	8.22	399	1979.05	3.13	2.43	406	1986.49	3.52	2.82	53	1964.40	2.37	1.67	1273	**1932.05**	0.68	0.00	1412	1958.98	2.09	1.39	617
123212	200	10	1776.56	1778.41	1788.70	0.68	0.58	199	1782.23	0.32	0.21	353	1786.79	0.58	0.47	34	1778.80	0.13	0.02	1398	1779.10	0.14	0.04	1314	**1778.41**	0.10	0.00	697
123222	200	20	1390.87	1390.87	1408.63	1.28	1.28	296	1396.24	0.39	0.39	530	1401.16	0.74	0.74	43	1453.82	4.53	4.53	2202	1396.42	0.40	0.40	1427	**1390.87**	0.00	0.00	518
Avg.						**3.52**	**3.34**	**330**		**1.85**	**1.68**	**421**		**1.27**	**1.10**	**43**		**1.51**	**1.34**	**1255**		**1.32**	**1.16**	**1371**		**0.57**	**0.40**	**595**
Global Avg.						**3.03**	**2.85**	**163**		**1.40**	**1.23**	**207**		**1.38**	**1.20**	**22**		**0.83**	**0.66**	**607**		**1.03**	**0.85**	**826**		**0.68**	**0.51**	**392**

Table 2.5: Detailed results for the second data subset DS2 (Prodhon Instances)

Instance	n	m	BKS	BKC	GRASP				MA|PM				LRGTS				GRASP + ELS				SALRP				2-Phase HGTS			
					Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time
20-5-1a	20	5	54793	54793	55021	0.42	0.42	0	**54793**	0.00	0.00	0	55131	0.62	0.62	1	**54793**	0.00	0.00	0	**54793**	0.00	0.00	20	**54793**	0.00	0.00	3
20-5-1b	20	5	39104	39104	**39104**	0.00	0.00	0	**39104**	0.00	0.00	0	**39104**	0.00	0.00	0	**39104**	0.00	0.00	0	**39104**	0.00	0.00	15	**39104**	0.00	0.00	4
20-5-2a	20	5	48908	48908	**48908**	0.00	0.00	0	**48908**	0.00	0.00	1	**48908**	0.00	0.00	1	**48908**	0.00	0.00	0	**48908**	0.00	0.00	19	48945	0.08	0.08	3
20-5-2b	20	5	37542	37542	**37542**	0.00	0.00	0	**37542**	0.00	0.00	0	**37542**	0.00	0.00	0	**37542**	0.00	0.00	0	**37542**	0.00	0.00	15	**37542**	0.00	0.00	4
Avg.						**0.10**	**0.10**	**0**		**0.00**	**0.00**	**0**		**0.15**	**0.15**	**1**		**0.00**	**0.00**	**0**		**0.00**	**0.00**	**17**		**0.02**	**0.02**	**4**
50-5-1a	50	5	90111	90111	90632	0.58	0.58	2	90160	0.05	0.05	4	90160	0.05	0.05	0	**90111**	0.00	0.00	3	**90111**	0.00	0.00	75	90402	0.32	0.32	27
50-5-1b	50	5	63242	63242	64761	2.40	2.40	2	**63242**	0.00	0.00	5	63256	0.02	0.02	1	**63242**	0.00	0.00	0	**63242**	0.00	0.00	58	64073	1.31	1.31	27
50-5-2a	50	5	88298	88298	88786	0.55	0.55	3	**88298**	0.00	0.00	5	88715	0.47	0.47	2	88643	0.39	0.39	11	**88298**	0.00	0.00	95	89342	1.18	1.18	23
50-5-2b	50	5	67308	67308	68042	1.09	1.09	3	67893	0.87	0.87	5	67698	0.58	0.58	2	**67308**	0.00	0.00	16	67340	0.05	0.05	59	68479	1.74	1.74	21
50-5-2bis	50	5	84055	84055	**84055**	0.00	0.00	2	**84055**	0.00	0.00	5	84181	0.15	0.15	3	**84055**	0.00	0.00	0	**84055**	0.00	0.00	75	**84055**	0.00	0.00	23
50-5-2bbis	50	5	51822	51822	52059	0.46	0.46	3	**51822**	0.00	0.00	6	51992	0.33	0.33	1	**51822**	0.00	0.00	11	**51822**	0.00	0.00	66	52087	0.51	0.51	29
50-5-3a	50	5	86203	86203	87380	1.37	1.37	3	**86203**	0.00	0.00	5	**86203**	0.00	0.00	0	**86203**	0.00	0.00	0	86456	0.29	0.29	74	**86203**	0.00	0.00	66
50-5-3b	50	5	61830	61830	61890	0.10	0.10	3	**61830**	0.00	0.00	8	**61830**	0.00	0.00	1	**61830**	0.00	0.00	0	62700	1.41	1.41	58	**61830**	0.00	0.00	38
Avg.						**0.82**	**0.82**	**3**		**0.12**	**0.12**	**5**		**0.20**	**0.20**	**1**		**0.05**	**0.05**	**5**		**0.22**	**0.22**	**70**		**0.63**	**0.63**	**32**
100-5-1a	100	5	275419	276186	279437	1.46	1.18	28	281944	2.37	2.08	33	277935	0.91	0.63	9	276960	0.56	0.28	148	277035	0.59	0.31	349	**276186**	0.28	0.00	157
100-5-1b	100	5	213615	214885	216159	1.19	0.59	24	216656	1.42	0.82	44	**214885**	0.59	0.00	9	215854	1.05	0.45	68	216002	1.12	0.52	269	214892	0.60	0.00	136
100-5-2a	100	5	193671	194124	199520	3.02	2.78	18	195568	0.98	0.74	45	196545	1.48	1.25	3	194267	0.31	0.07	212	**194124**	0.23	0.00	349	194625	0.49	0.26	145
100-5-2b	100	5	157150	157150	159550	1.53	1.53	23	157325	0.11	0.11	45	157792	0.41	0.41	4	157375	0.14	0.14	125	**157150**	0.00	0.00	212	157319	0.11	0.11	193
100-5-3a	100	5	200079	200242	203999	1.96	1.88	21	201749	0.83	0.75	36	201952	0.94	0.85	3	200345	0.13	0.05	141	**200242**	0.08	0.00	250	201086	0.50	0.42	163
100-5-3b	100	5	152441	152467	154596	1.41	1.40	20	153322	0.58	0.56	43	154709	1.49	1.47	3	152528	0.06	0.04	221	**152467**	0.02	0.00	197	153663	0.80	0.78	168
Avg.						**1.76**	**1.56**	**22**		**1.05**	**0.85**	**41**		**0.97**	**0.77**	**5**		**0.37**	**0.17**	**153**		**0.34**	**0.14**	**271**		**0.46**	**0.26**	**160**
100-10-1a	100	10	287983	289755	323171	12.22	11.53	38	316575	9.93	9.26	31	291887	1.36	0.74	14	301418	4.67	4.03	48	291043	1.06	0.44	270	**289755**	0.62	0.00	277
100-10-1b	100	10	231763	234210	271477	17.14	15.91	30	270251	16.61	15.39	45	235532	1.63	0.56	14	269594	16.32	15.11	186	**234210**	1.06	0.00	203	238002	2.69	1.62	152
100-10-2a	100	10	243590	243778	254087	4.31	4.23	39	245123	0.63	0.55	31	246708	1.28	1.20	15	**243778**	0.08	0.00	260	245813	0.91	0.83	261	245768	0.89	0.82	92
100-10-2b	100	10	203988	203988	206555	1.26	1.26	30	205052	0.52	0.52	39	204435	0.22	0.22	10	**203988**	0.00	0.00	139	205312	0.65	0.65	199	204252	0.13	0.13	99
100-10-3a	100	10	250882	250882	270826	7.95	7.95	35	253669	1.11	1.11	36	258656	3.10	3.10	14	253511	1.05	1.05	164	**250882**	0.00	0.00	338	254716	1.53	1.53	125
100-10-3b	100	10	204317	204815	216173	5.80	5.55	40	**204815**	0.24	0.00	45	205883	0.77	0.52	11	205087	0.38	0.13	203	205009	0.34	0.09	240	205837	0.74	0.50	144
Avg.						**8.11**	**7.74**	**35**		**4.84**	**4.47**	**38**		**1.39**	**1.06**	**13**		**3.75**	**3.39**	**167**		**0.67**	**0.34**	**252**		**1.10**	**0.77**	**148**
200-10-1a	200	10	476778	476778	490820	2.95	2.95	518	483497	1.41	1.41	431	481676	1.03	1.03	63	486467	2.03	2.03	1521	481002	0.89	0.89	1428	**476778**	0.00	0.00	671
200-10-1b	200	10	378289	378289	416753	10.17	10.17	379	380044	0.46	0.46	579	380613	0.61	0.61	60	382329	1.07	1.07	359	383586	1.40	1.40	1336	**378289**	0.00	0.00	476
200-10-2a	200	10	449849	449951	512679	13.97	13.94	554	451840	0.44	0.42	351	453353	0.78	0.76	60	452276	0.54	0.52	112	450848	0.22	0.20	1796	**449951**	0.02	0.00	483
200-10-2b	200	10	374330	374961	379980	1.51	1.34	368	375019	0.18	0.02	401	377351	0.81	0.64	78	376027	0.45	0.28	1610	376674	0.63	0.46	1245	**374961**	0.17	0.00	530
200-10-3a	200	10	472321	472321	496694	5.16	5.16	425	478132	1.23	1.23	266	476684	0.92	0.92	78	478380	1.28	1.28	1596	473875	0.33	0.33	1776	**472321**	0.00	0.00	624
200-10-3b	200	10	362817	363252	389016	7.22	7.09	290	364834	0.56	0.44	341	365250	0.67	0.55	74	365166	0.65	0.53	591	363701	0.24	0.12	1326	**363252**	0.12	0.00	389
Avg.						**6.83**	**6.77**	**422**		**0.71**	**0.66**	**395**		**0.80**	**0.75**	**69**		**1.00**	**0.95**	**965**		**0.62**	**0.57**	**1484**		**0.05**	**0.00**	**529**
Global Avg.						**3.57**	**3.45**	**97**		**1.35**	**1.23**	**96**		**0.71**	**0.59**	**18**		**1.04**	**0.92**	**258**		**0.38**	**0.27**	**422**		**0.49**	**0.38**	**176**

Table 2.6: Detailed results for the third data subset DS3 (Barreto Instances)

Instance	n	m	BKS	BKC	GRASP				MA|PM				LRGTS				GRASP + ELS				SALRP				2-Phase HGTS			
					Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time	Cost	Gap BKS	Gap BKC	CPU time
Christofides69-50x5	50	5	565.6	565.6	599.1	5.92	5.92	3	**565.6**	0.00	0.00	4	586.4	3.68	3.68	3	**565.6**	0.00	0.00	8	**565.6**	0.00	0.00	53	580.4	2.62	2.62	45
Christofides69-75x10	75	10	844.4	848.9	861.6	2.04	1.50	10	866.1	2.57	2.03	9	863.5	2.26	1.72	10	850.8	0.76	0.22	86	**848.9**	0.53	0.00	127	**848.9**	0.53	0.00	94
Christofides69-100x10	100	10	833.4	833.4	861.6	3.38	3.38	26	850.1	2.00	2.00	45	842.9	1.14	1.14	28	**833.4**	0.00	0.00	127	838.3	0.59	0.59	331	838.6	0.62	0.62	234
Daskin95-88x8	88	8	355.8	355.8	356.9	0.31	0.31	18	**355.8**	0.00	0.00	34	368.7	3.63	3.63	18	**355.8**	0.00	0.00	130	**355.8**	0.00	0.00	577	362.0	1.74	1.74	148
Daskin95-150x10	150	10	43919.9	43963.6	44625.2	1.61	1.50	156	44011.7	0.21	0.11	255	44386.3	1.06	0.96	119	**43963.6**	0.10	0.00	1697	45109.4	2.71	2.61	323	44578.9	1.50	1.40	456
Gaskell67-21x5	21	5	424.9	424.9	429.6	1.11	1.11	0	**424.9**	0.00	0.00	0	**424.9**	0.00	0.00	0	**424.9**	0.00	0.00	0	**424.9**	0.00	0.00	18	**424.9**	0.00	0.00	6
Gaskell67-22x5	22	5	585.1	585.1	**585.1**	0.00	0.00	0	611.8	4.56	4.56	0	587.4	0.39	0.39	0	**585.1**	0.00	0.00	15	**585.1**	0.00	0.00	17	**585.1**	0.00	0.00	9
Gaskell67-29x5	29	5	512.1	512.1	515.1	0.59	0.59	0	**512.1**	0.00	0.00	1	**512.1**	0.00	0.00	0	**512.1**	0.00	0.00	9	**512.1**	0.00	0.00	24	**512.1**	0.00	0.00	11
Gaskell67-32x5	32	5	562.2	562.2	571.9	1.73	1.73	1	571.9	1.73	1.73	1	584.6	3.98	3.98	1	**562.2**	0.00	0.00	18	**562.2**	0.00	0.00	27	**562.2**	0.00	0.00	40
Gaskell67-32x5	32	5	504.3	504.3	**504.3**	0.00	0.00	1	534.7	6.03	6.03	1	504.8	0.10	0.10	1	**504.3**	0.00	0.00	34	**504.3**	0.00	0.00	25	**504.3**	0.00	0.00	22
Gaskell67-36x5	36	5	460.4	460.4	**460.4**	0.00	0.00	1	485.4	5.43	5.43	1	476.5	3.50	3.50	1	**460.4**	0.00	0.00	0	**460.4**	0.00	0.00	32	**460.4**	0.00	0.00	39
Min92-27x5	27	5	3062.0	3062.0	**3062.0**	0.00	0.00	0	**3062.0**	0.00	0.00	1	3065.2	0.10	0.10	0	**3062.0**	0.00	0.00	35	**3062.0**	0.00	0.00	23	**3062.0**	0.00	0.00	11
Min92-134x8	134	8	5709.0	5709.0	5965.1	4.49	4.49	50	5950.0	4.22	4.22	111	5809.0	1.75	1.75	48	5719.3	0.18	0.18	280	**5709.0**	0.00	0.00	522	5890.6	3.18	3.18	252
Global Avg.						**1.63**	**1.58**	**20**		**2.06**	**2.01**	**36**		**1.66**	**1.61**	**18**		**0.08**	**0.03**	**188**		**0.29**	**0.25**	**161**		**0.78**	**0.74**	**105**

Chapter 3

A comparison of heuristic algorithms for the CLRP

Notes about the chapter

The contents of this chapter is based on the paper entitled "*A computational comparison of heuristic algorithms for the capacitated location-routing problem*", co-authored with Rodrigo Linfati, Professor Maria Gulnara Baldoquin and Professor Paolo Toth, which has been submitted for publication. Partial results have been presented in the 5th International Workshop on Freight Transportation and Logistics (ODYSSEUS 2012), Mykonos–Greece, in the first meeting of the EURO Working Group on Vehicle Routing and Logistics Optimization (VEROLOG 2012), Bologna– Italy, and in the INFORMS Annual Meeting 2012, Phoenix – USA.

3.1 Introduction

In this work, we propose two new heuristics, and present a computational comparative study of the most effective heuristics proposed for the CLRP. The new algorithms use the initialization procedure and the neighborhood structures introduced for algorithm 2-Phase HGTS in Escobar et al. [19]. We compare the results of the proposed algorithms with the algorithm explained in Chapter 2 *(Algorithm 2-Phase HGTS)* to obtain the best performing algorithm. The first new algorithm, called *Granular Variable Tabu Neighborhood Search* (GTVNS), considers a *Variable Neighborhood Search* (VNS) procedure, that includes a *Granular Tabu Search* approach, to enhance the quality of solution S_0. The second new algorithm, called *Granular Simulated Annealing* (GSA), considers a *Simulated Annealing* (SA) method, with a granular search space, to improve solution S_0.

The main contribution of the chapter is the development of an effective heuristic algorithm, called GTVNS, for the solution of the CLRP. The algorithm exploits the systematic changes of the neighborhood structures and the neighborhood topologies considered in the *Variable Neighborhood Search* (VNS) scheme to guide a trajectory local search procedure according to the *Granular Tabu Search* (GTS) approach. The proposed algorithm is a novel metaheuristic approach which combines VNS with GTS techniques for getting good results within short computing times. While a combination between VNS and Tabu Search (TS) has been proposed in the literature (see e.g. Moreno Pérez et al. [39] and Repoussis et al. [53]), no attempt has been proposed for combining a GTS technique within a VNS scheme. The basic VNS scheme some times meets difficulties to escape from local optima, while the GTS approach has no such difficulties, since infeasible solutions are allowed, and the memory technique prevents cycling, allowing the algorithm to escape from local optima.

3.2 General framework

3.2.1 Granular search space

The granular search approach, proposed in Toth and Vigo [60], is based on the utilization of a sparse graph containing the edges incident to the depots, the edges belonging to the best solutions found so far, and the edges whose cost is smaller than a granularity threshold $\vartheta = \beta\bar{z}$, where $\bar{z}$ is the average cost of the edges in the best solution found so far, and β is a sparsification factor which is dynamically updated during the search. The main idea of the granularity approach is to obtain high quality solutions within short computing times. To evaluate this significant effect, a computational comparison of the considered algorithms is performed, by executing them with and without the granular search approach.

3.2.2 Neighborhood structures

The considered heuristics use *intra-route moves* (performed in the same route) and *inter-route moves* (performed between two routes assigned to the same depot or to different depots) corresponding to five neighborhood structures $N_k(k = 1, ..., 5)$: described in Chapter 2. A move is performed only if all the new edges inserted in the solution are in the "granular" search space. Finally, the shaking procedure described in Chapter 2 is not used in algorithms GTVNS and GSA.

3.2.3 Initial solution

The initial solution S_0 is constructed by using a hybrid heuristic, proposed in Escobar et al. [19]and based on a cluster approach, which is able to find good initial feasible solutions within short computing times. In order to make a comparative study, a "good" and a "bad" initial solutions are chosen to initialize the three algorithms. The "good" and the "bad" initial solutions are obtained by executing the splitting procedure "many" and "few" times, respectively.

3.3 Description of the new proposed algorithms

3.3.1 The Granular Variable Tabu Neighborhood Search heuristic algorithm (GTVNS)

The GTVNS algorithm combines the potentiality of the systematic changes of neighborhood structures proposed by Mladenović and Hansen [38] and the efficient Granular Tabu Search (GTS) approach introduced by Toth and Vigo [60]. The Variable Neighborhood Search (VNS) is a metaheuristic approach which applies a search strategy based on the systematic change of the neighborhood structures to escape from local optima. Three main elements are considered during the systematic change of the neighborhoods: (1) A local minimum with respect to a given neighborhood is not necessarily the same for the other neighborhoods; (2) A global minimum is a local minimum for all the possible neighborhood structures; (3) Local minima with respect to the neighborhood structures should be relatively close each other. In the proposed algorithm, the VNS technique controls the neighborhood changes, while the GTS technique guides the search process by using the neighborhood structures and the efficient search space detailed in the previous sub chapters. After constructing the initial solution S_0, the VNS procedure iterates through different neighborhood structures to improve the best feasible solution (S^*) found so far. The algorithm starts by setting $S^\star = \overline{S} = \widehat{S} = S_0$, where $\overline{S}$ is the current (feasible or infeasible) solution, and $\widehat{S}$ is the current feasible solution. The following steps are then repeated until a stopping criterion (number of iterations or computing time) is reached:

1. Select a neighborhood from the neighborhoods structures $N_k(k = 1, ..., 5)$;
2. Local search: apply a Granular Tabu Search (GTS) procedure in the selected neighborhood $N_k\left(\overline{S}\right)$ until a local minimum S' is found;
3. If S' is infeasible and $F_2\left(S'\right) \leq F_2\left(\overline{S}\right)$, set $\overline{S}:= S'$;
4. If S' is feasible and $F_1\left(S'\right) \leq F_1\left(\widehat{S}\right)$, set $\widehat{S} := S'$ and $\overline{S}:= S'$;
5. Every $N_g \times n$ iterations apply the third diversification strategy used by algorithm 2-Phase HGTS.

Finally, the best feasible solution found so far $S^\star$ is kept. The GTS procedure explores the solution space by moving, at each iteration, from a solution $\overline{S}$ to the best solution S in the neighborhood $N\left(\overline{S}\right)$. The best possible move is selected as the move in $N\left(\overline{S}\right)$ producing the smallest value of the objective function $F_2\left(S\right)$ and of the following tabu aspiration criterion: if the value of the objective function $F_1\left(S\right)$ of the new solution S is not greater than the cost of the best solution found so far, the move producing S is performed even if it corresponds to *tabu move*.

3.3.2 The Granular Simulated Annealing heuristic algorithm (GSA)

The GSA algorithm considers a standard implementation of the Simulated Annealing metaheuristic (SA) with a reduced local search space. Let $S^\star$ be the best feasible solution found so far, $\overline{S}$ the current solution (feasible or infeasible), $\widehat{S}$ the current feasible solution, α the *cooling factor*, and T

the current *temperature*. Initially, we set $S^{\star}:= S_0$, $\overline{S}:= S_0$, and $\widehat{S} := S_0$. In addition, we determine the initial temperature T_0 (where T_0 is a given parameter), and set i:=0. The proposed algorithm performs the following steps until a stopping criterion (number of iterations or computing time) is met:

1. Every N_{cool} iterations (where N_{cool} is a given parameter) set i:=$i+1$, and decrease the current temperature T according to the function $T = T_i = \theta T_{i-1}$, where $0 < \theta < 1$ (with θ given parameter);
2. Generate a random solution S' in the union of the neighborhoods of the current solution $\overline{S}$ obtained by considering all the neighborhood structures $N_k(k = 1, ..., 5)$;
3. Compute $\sigma = F_2(S') - F_2(\overline{S})$;
4. Generate a random number r in the range $[0, 1]$;
5. If $\sigma \leq 0$ do:
   (a) If S' is feasible, set $\overline{S}:= S'$, $\widehat{S}:= S'$;
   (b) If S' is infeasible, set $\overline{S}:= S'$;
6. If $\sigma > 0$ do:
   (a) If $r < \exp(-\sigma/T)$ and S' is feasible, set $\overline{S}:= S'$ and $\widehat{S}:=S'$;
   (b) If $r < \exp(-\sigma/T)$ and S' is infeasible, set $\overline{S}:=S'$;

Finally, the best feasible solution found so far $S^{\star}$ is kept.

3.4 Computational experiments

The comparison of the effects of the initial solution and of the granularity approach on the performance of algorithms 2-Phase HGTS, GTVNS and GSA has been performed by fixing, for each instance, the same maximum CPU time as stopping criterion. In particular, the CPU time for each instance has been defined as the maximum among the CPU times spent by the three considered algorithms, each using its "best" initial solution and the parameter values detailed in Subsection 3.4.2, to solve the given instance.

After having defined, for each of the three considered algorithms, the corresponding best configuration with respect to the initial solution and the utilization of the granularity approach, the best performance of each algorithm has been evaluated by executing $N_{stop} \times n$ iterations (where N_{stop} is a given parameter) for each instance. After extensive computational tests, we have determined that the best values of N_{stop} are 10, 7 and 6000 for algorithms 2-Phase HGTS, GTVNS and GSA, respectively. For each considered instance, algorithm GSA has been run five times with different random generator seeds. The results reported in Tables 3.1 to 3.4 for algorithm GSA correspond, for each instance, to the best solution value obtained over the five runs with the corresponding total running time of the algorithm. Algorithm GTVNS is a "deterministic" algorithm, and, for each instance, a single run has been executed. The implementation details and the results are discussed in the following subsections.

3.4.1 Implementation details

The three described algorithms have 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 algorithms have been evaluated by considering 79 benchmark instances from the literature. The complete set of instances considers three data subsets proposed by Tuzun and Burke [61], Prins et al. [45] (called "Prodhon Instances" in the following), and Barreto [7]. In all the subsets, the customers and the depots are represented by points in the plane. Consequently, the traveling cost of an edge is the Euclidean distance, multiplied by 100 and rounded up to the next integer (Prins et al. [45]), or calculated as a double-precision real number (Tuzun and Burke [61]and Barreto [7]).

The first data subset was proposed by Tuzun and Burke [61], and contains 36 instances with uncapacitated depots. The number of customers is $n = 100, 150$ and 200. The number of potential depots is either 10 or 20. The vehicle capacity is set to 150. The second data subset was introduced by Prins et al. [45], and considers 30 instances with capacity constraints on routes and depots. The number of customers is $n = 20, 50, 100$ and 200. The number of potential depots is either 5 or 10. The vehicle capacity is either 70 or 150. Finally, the third data subset is proposed by Barreto [7], and considers 13 instances obtained by modifying some classical CVRP instances by adding new depots with capacities and fixed costs. The number of customers ranges from 21 to 150, and the number of potential depots from 5 to 10.

3.4.2 Parameter settings

A suitable set of parameters, whose values are based on extensive computational tests on the benchmark instances, was selected for each algorithm and is reported in the following:

	2-Phase HGTS	**GTVNS**	**GSA**
β_o	1.50	1.80	1.50
β_n	2.40	2.40	2.50
N_s	2.00	2.00	2000
N_r	1.00	1.00	1000
N_{fact}	10	10	10
γ_d	0.0075	0.0075	-
γ_r	0.0100	0.0050	-
γ_{min}	$1/F_1(S_0)$	$1/F_1(S_0)$	-
γ_{max}	0.0400	0.0400	-
δ_{red}	0.30	0.30	-
δ_{inc}	2.00	2.00	-
h	0.02	0.01	-
N_{shake}	0.20	-	-
N_g	1.50	1.50	-
t_{min}	5	3	-
t_{max}	10	8	-
N_{cool}	-	-	1200
θ	-	-	0.90
T_0	-	-	1000

These values have been utilized for the comparison of the solutions obtained by the three described algorithms.

3.4.3 Comparison of the three described algorithms

We first compare the performance of the algorithms described in Sub chapter 3.3 with the algorithm proposed in Chapter 2 (2-Phase HGTS), by considering the different configurations obtained by starting with a "good" or a "bad" solutions, and by applying or not the granularity approach. Then, for the three algorithms, we consider the corresponding best configurations, and compare them in order to determine the best performing algorithm. The best algorithm is finally compared with the most effective heuristic algorithms proposed in the literature for the solution of the CLRP: GRASP+ELS of Duhamel et al. [18], SALRP of Yu et al. [66], ALNS of Hemmelmayr et al. [29], GRASP+ILP of Contardo et al. [13], and MACO of Ting and Chen [59].

In Tables 3.1 to 3.9, the following notation is used:

Instance	instance name;
Cost	solution cost obtained by the corresponding algorithm in one single run;
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;
PBKS	cost of the previous best-known solution given by the minimum cost among those found by algorithms GRASP+ELS, SALRP, ALNS-500K, ALNS-5000K, GRASP+ILP, and MACO;
BKS	cost of the best known solution = min {PBKS, solution cost found by the proposed algorithms};
NBKS	number of best results (BKS) obtained by the corresponding algorithm;
NIBS	number of instances for which the corresponding algorithm is the only one which found BKS;
CPU	CPU used by the corresponding algorithm;
CPU index	Passmark performance test for the corresponding CPU;
CPU time	running time in seconds on the CPU used by the corresponding algorithm;
Gap PBKS	percentage gap of the solution cost found by the corresponding algorithm in one single run with respect to PBKS;

Gap Best PBKS	percentage gap of the best solution cost found by the corresponding algorithm over the executed runs with respect to PBKS;
Gap Avg. PBKS	percentage gap of the average solution cost found by the corresponding algorithm over the executed runs with respect to PBKS.

In addition, for each instance, the costs which are equal to the corresponding BKS are reported in bold. Whenever the considered algorithm is the only one which found the corresponding BKS value, the reported cost is underlined. Finally, the CPU index of a CPU is given by the Passmark performance test (for further details see [1]). This is a well known benchmark test focused on CPU and memory performance. A higher value of the CPU index indicates that the corresponding CPU is faster.

3.4.4 Comparison of the effect of the initial solution

The performance of the three algorithms is first compared by considering two different initial solutions. Let G_0 denote a "good" initial solution and B_0 a "bad" initial solution. Solutions G_0 and B_0 are determined by executing the splitting procedure for 7 and 3 iterations respectively.

Table 3.1 shows the summarized results corresponding to the average values of Gap PBKS and of the CPU times by starting from solutions G_0 and B_0. The results show that GSA is not highly sensitive to the quality of the initial solution, while 2-Phase HGTS provides the best global average results by using the initial solution G_0. Finally, GTVNS obtains the best average results by using the initial solution B_0. In the following, we will consider, as initial solution, G_0 for algorithms 2-Phase HGTS and GSA, and B_0 for algorithm GTVNS.

3.4.5 Comparison of the effect of the granularity

We consider now the impact of the granularity approach on the performance of the three algorithms. These results are summarized in Table 3.2. It is to note that GTVNS and 2-Phase HGTS provide an equivalent global performance when executed without the "granular" search approach. The results show that the granular search approach significantly improves the performance of the three algorithms, hence, in the following we will consider this configuration for all the algorithms.

3.4.6 Global comparison

Tables 3.3, 3.4 and 3.5 provide the detailed results of the three algorithms on the three data sets Tuzun-Burke, Prodhon and Barreto, respectively. The results clearly show that algorithm GTVNS outperforms the other two algorithms for what concerns both the CPU time and the quality of the solutions found. Indeed, for all the data sets, the average value of Gap PBKS, and the values of NBKS and NIBS of algorithm GTVNS are better than the corresponding values of algorithms 2-Phase HGTS and GSA. In addition, by considering all the 79 instances of the three data sets, algorithms GTVNS finds, with respect to algorithm 2-Phase HGTS, 45 better solutions and 7 worse solutions, and with respect to algorithm GSA, 58 better solutions and only 1 worse solution. Therefore algorithm GTVNS is the best performing of the three described algorithms, and, in the following section, it will be compared with the most effective heuristics from the literature.

3.4.7 Comparison of the most efficient algorithms

In Tables 3.6 to 3.9, we compare algorithm GTVNS with the most effective heuristics proposed for the solution of the CLRP, i.e., as previously mentioned, algorithms GRASP+ELS of Duhamel et al. [18], SALRP of Yu et al. [66], ALNS of Hemmelmayr et al. [29], GRASP+ILP of Contardo et al. [13], and MACO of Ting and Chen [59]. In the tables, we report the results as presented in the corresponding papers.

Algorithm GRASP+ELS has been executed five times and only the best solutions found over the five runs are reported. In addition, it is to note that the CPU time reported for each instance represents the time required to find the best solution within the corresponding run. The results reported for algorithm SALRP correspond to a single run of the algorithm. For algorithm ALNS, the best and the average costs over five runs for 500K iterations

(ALNS - 500K), as well as the best costs over five runs for 5000K iterations (ALNS - 5000K), are reported. The CPU time reported for each instance corresponds to the total running time of the corresponding algorithm. Algorithms GRASP+ILP and MACO have been executed for ten runs. The results reported for algorithm GRASP+ILP correspond, for each instance, to the best and to the average costs found, and to the average CPU time over the ten runs. The results reported for algorithm MACO correspond to the best cost found and to the average CPU time over the ten runs. Finally, the results reported for algorithm GTVNS correspond to a single run of the algorithm.

Table 3.6 shows a summary of the results found by the algorithms on the complete data set, while Tables 3.7 to 3.9 show the detailed results for the three considered data sets. 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 GRASP+ILP and MACO 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.

As shown in Table 3.6, for what concerns the global average value of Gap PBKS, algorithm GTVNS obtains better results than those obtained by algorithms GRASP+ELS, SALRP and MACO. In addition, by considering the global average value of the gaps corresponding to the average costs computed over several runs (Gap Avg. PBKS), Table 3.6 shows that algorithm GTVNS obtains results better than those obtained (in comparable CPU times) by algorithm ALNS-500K, and slightly worse than those obtained (in much larger CPU times) by algorithm GRASP+ILP. The best results on the global average value of Gap Best PBKS are obtained, with very large CPU times, by algorithms GRASP+ILP and ALNS-5000K. 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 GTNVS with respect to the best results reported for algorithms GRASP+ILP and ALNS-5000K.

For what concerns the number NBKS of the 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, algorithms ALNS-5000K and GRASP+ILP are again the best ones, while algorithms ALNS-500K (Best solution) and GTVNS have comparable behaviors (although the former algorithm has larger CPU times). Finally, it is to note that algorithm GTVNS is able to find, within short CPU times, 28 best known solutions and to improve the previous best known solution for 5 instances.

As for the global CPU time, algorithm GTVNS is faster than the previous published algorithms which are able to find the best results in terms of average gaps and number of best known solutions. Algorithm MACO seems to require smaller CPU times than algorithm GTVNS, but since only the average computing times over ten runs are reported for the former algorithm, instead of the complete running times for executing the ten runs, a comparison between the two algorithms may be biased.

3.5 Concluding remarks

The computational experiments show that algorithm GTVNS generally obtains better results, in terms of average Gap BKS, NBKS and NIBS, than those obtained by algorithms 2-Phase HGTS and GSA. The results emphasize the importance of the granular search approach for the three considered algorithms, by showing that it significantly improves the performance of algorithms GTVNS and 2-Phase HGTS. We have also compared the performance of algorithm GTVNS with that of the most recent effective published heuristics for the CLRP on a set of benchmarking instances from the literature. The results show the effectiveness of algorithm GTVNS, which is able to improve some best known results within short computing times.

Table 3.1: Summarized results on Gap PBKS by comparingthe quality of the Initial Solutions (G0 and B0)

Set	Size	Initial Solution (G_0)	2-Phase HGTS (G_0)	GTVNS (G_0)	GSA (G_0)	Average CPU Time	Initial Solution (B_0)	2-Phase HGTS (B_0)	GTVNS (B_0)	GSA (B_0)	Average CPU Time
		Average Gap PBKS	Average Gap PBKS	Average Gap PBKS	Average Gap PBKS		Average Gap PBKS	Average Gap PBKS	Average Gap PBKS	Average Gap PBKS	
Tuzun-Burke	36	1.55	1.00	1.32	1.13	592	1.81	1.28	0.62	1.25	592
Prodhon	30	1.33	0.54	1.15	1.21	265	1.48	0.79	0.34	1.24	265
Barreto	13	1.74	0.78	0.97	1.33	160	2.04	0.96	0.66	1.33	160
Total	79										
Global Avg.		**1.49**	**0.79**	**1.20**	**1.20**	**397**	**1.72**	**1.04**	**0.52**	**1.26**	**397**

Table 3.2: Summarized results on Gap PBKSwithout the "granular" search approach

Set	**Size**	**2-Phase HGTS (G_0)**	**GTVNS (B_0)**	**GSA (G_0)**	**Average CPU Time**
		Average Gap PBKS	**Average Gap PBKS**	**Average Gap PBKS**	
Tuzun-Burke	36	0.96	0.93	1.33	592
Prodhon	30	0.88	0.78	1.26	265
Barreto	13	0.87	1.11	1.68	160
Total	79				
Global Avg.		**0.91**	**0.90**	**1.36**	**397**

Table 3.3: Best results for 2-Phase HGTS, GTVNS,and GSA on Tuzun-Burke Instances

Instance	PBKS	2-Phase HGTS			GTVNS			GSA		
		Cost	Gap PBKS	CPU time	Cost	Gap PBKS	CPU time	Cost	Gap Best PBKS	CPU time
111112	**1467.68**	1479.21	0.79	152	1479.21	0.79	84	1490.82	1.58	151
111122	**1449.20**	1486.27	2.56	239	1485.28	2.49	126	1486.27	2.56	244
111212	**1394.80**	1407.26	0.89	120	1402.59	0.56	74	1407.26	0.89	123
111222	**1432.29**	1474.01	2.91	146	1463.23	2.16	99	1474.01	2.91	159
112112	**1167.16**	**1167.16**	0.00	232	**1167.16**	0.00	83	**1167.16**	0.00	246
112122	**1102.24**	**1102.24**	0.00	224	**1102.24**	0.00	105	**1102.24**	0.00	220
112212	**791.66**	**791.66**	0.00	201	**791.66**	0.00	96	**791.66**	0.00	181
112222	**728.30**	**728.30**	0.00	254	**728.30**	0.00	126	**728.30**	0.00	254
113112	**1238.49**	**1238.49**	0.00	160	**1238.49**	0.00	82	**1238.49**	0.00	157
113122	**1245.31**	1251.22	0.47	237	1247.27	0.16	127	1251.22	0.47	242
113212	**902.26**	**902.26**	0.00	135	**902.26**	0.00	71	**902.26**	0.00	137
113222	**1018.29**	**1018.29**	0.00	157	**1018.29**	0.00	85	**1018.29**	0.00	159
131112	**1914.41**	1961.75	2.47	485	1933.67	1.01	179	1944.57	1.58	353
131122	**1823.20**	1856.51	1.83	298	1852.14	1.59	173	1871.13	2.63	273
131212	**1969.80**	2012.69	2.18	406	1983.09	0.67	184	2012.69	2.18	411
131222	**1792.80**	1803.01	0.57	302	1803.01	0.57	175	1803.01	0.57	263
132112	**1444.73**	1445.25	0.04	449	**1443.32**	-0.10	186	1453.78	0.63	446
132122	**1434.63**	1452.07	1.22	493	1441.43	0.47	210	1452.07	1.22	491
132212	**1204.42**	**1204.42**	0.00	270	**1204.42**	0.00	128	**1204.42**	0.00	269
132222	**931.28**	931.49	0.02	335	**931.28**	0.00	177	931.44	0.02	320
133112	**1694.18**	1705.36	0.66	444	1701.34	0.42	182	1745.23	3.01	425
133122	**1392.01**	1416.74	1.78	342	1416.74	1.78	175	1416.74	1.78	333
133212	**1198.20**	1234.83	3.06	526	1213.87	1.31	207	1234.83	3.06	497
133222	**1151.80**	1156.05	0.37	380	**1151.80**	0.00	208	1156.27	0.39	371
121112	**2249.00**	2265.59	0.74	522	2258.02	0.40	315	2265.59	0.74	503
121122	**2153.80**	2166.43	0.59	603	2166.20	0.58	300	2187.86	1.58	526
121212	**2212.40**	2249.40	1.67	527	2239.65	1.23	287	2256.32	1.99	486
121222	**2230.94**	2237.81	0.31	558	2236.73	0.26	351	2253.32	1.00	506
122112	**2073.73**	2121.93	2.32	522	2103.82	1.45	278	2121.93	2.32	474
122122	**1692.17**	1749.10	3.36	691	1717.92	1.52	433	1718.65	1.56	605
122212	**1453.18**	1473.27	1.38	724	1469.45	1.12	318	1473.27	1.38	747
122222	**1082.74**	1082.59	-0.01	616	**1082.46**	-0.03	349	1082.99	0.02	578
123112	**1960.30**	1984.77	1.25	542	1969.38	0.46	261	1990.87	1.56	475
123122	**1926.64**	1958.98	1.68	617	1935.74	0.47	344	1970.91	2.30	586
123212	**1762.03**	1778.41	0.93	697	1776.90	0.84	349	1779.10	0.97	533
123222	**1391.68**	1390.87	-0.06	518	1391.50	-0.01	317	**1390.74**	-0.07	489
Average		**1519.05**	**1.00**	**392**	**1512.50**	**0.62**	**201**	**1521.55**	**1.13**	**368**
NBKS		**8**			**12**			**9**		
NIBS		**0**			**2**			**1**		

Table 3.4: Best results for 2-Phase HGTS, GTVNS,and GSA on Prodhon Instances

Instance	PBKS	2-Phase HGTS			GTVNS			GSA		
		Cost	Gap PBKS	CPU time	Cost	Gap PBKS	CPU time	Cost	Gap Best PBKS	CPU time
20-5-1a	**54793**	**54793**	0.00	3	**54793**	0.00	2	**54793**	0.00	4
20-5-1b	**39104**	**39104**	0.00	4	**39104**	0.00	3	39253	0.38	4
20-5-2a	**48908**	48945	0.08	3	48945	0.08	2	50570	3.40	4
20-5-2b	**37542**	**37542**	0.00	4	**37542**	0.00	3	37611	0.18	4
50-5-1a	**90111**	90402	0.32	27	**90111**	0.00	13	92413	2.55	30
50-5-1b	**63242**	64073	1.31	27	**63242**	0.00	9	65002	2.78	25
50-5-2a	**88298**	89342	1.18	23	89342	1.18	12	89342	1.18	22
50-5-2b	**67308**	68479	1.74	21	67951	0.96	10	68771	2.17	22
50-5-2bis	**84055**	**84055**	0.00	23	84126	0.08	8	84911	1.02	22
50-5-2bbis	**51822**	52087	0.51	29	52213	0.75	9	52270	0.86	17
50-5-3a	**86203**	**86203**	0.00	66	**86203**	0.00	18	86957	0.87	37
50-5-3b	**61830**	**61830**	0.00	38	61885	0.09	20	62902	1.73	39
100-5-1a	**275406**	276186	0.28	157	276137	0.27	75	278991	1.30	131
100-5-1b	**213704**	214892	0.56	136	216154	1.15	59	216668	1.39	116
100-5-2a	**193671**	194625	0.49	145	193896	0.12	76	194941	0.66	149
100-5-2b	**157095**	157319	0.14	193	157180	0.05	82	157319	0.14	178
100-5-3a	**200242**	201086	0.42	163	200777	0.27	69	204392	2.07	137
100-5-3b	**152441**	153663	0.80	168	153435	0.65	68	153663	0.80	141
100-10-1a	**288415**	289755	0.46	277	**287864**	-0.19	203	289755	0.46	280
100-10-1b	**230989**	238002	3.04	152	232599	0.70	117	238903	3.43	147
100-10-2a	**243695**	245768	0.85	92	245484	0.73	52	245768	0.85	81
100-10-2b	**203988**	204252	0.13	99	204252	0.13	42	204979	0.49	84
100-10-3a	**250882**	254716	1.53	125	254558	1.47	82	256267	2.15	121
100-10-3b	**204601**	205837	0.60	144	205824	0.60	78	208993	2.15	110
200-10-1a	**475344**	476778	0.30	671	477009	0.35	320	477619	0.48	552
200-10-1b	**377043**	378289	0.33	476	377716	0.18	239	378289	0.33	450
200-10-2a	**449152**	449951	0.18	483	**449006**	-0.03	231	450578	0.32	480
200-10-2b	**374469**	374961	0.13	530	374717	0.07	290	378456	1.06	411
200-10-3a	**469706**	472321	0.56	624	471978	0.48	330	472380	0.57	530
200-10-3b	**362743**	363252	0.14	389	362827	0.02	214	364931	0.60	318
Average		**197617**	**0.54**	**176**	**197229**	**0.34**	**91**	**198590**	**1.21**	**155**
NBKS		**6**			**8**			**1**		
NIBS		**0**			**2**			**0**		

Table 3.5: Results for 2-Phase HGTS, GTVNS,and GSA on Barreto Instances

Instance	PBKS	2-Phase HGTS			GTVNS			GSA		
		Cost	Gap PBKS	CPU time	Cost	Gap PBKS	CPU time	Cost	Gap Best PBKS	CPU time
Christofides69-50x5	**565.6**	580.4	2.62	45	580.4	2.62	22	588.3	4.01	46
Christofides69-75x10	**844.4**	848.9	0.53	94	853.8	1.11	45	868.9	2.90	91
Christofides69-100x10	**833.4**	838.6	0.62	234	837.1	0.44	111	841.7	1.00	220
Daskin95-88x8	**355.8**	362.0	1.74	148	361.6	1.63	97	368.4	3.54	152
Daskin95-150x10	**43963.6**	44578.9	1.40	456	44578.9	1.40	199	44881.8	2.09	399
Gaskell67-21x5	**424.9**	**424.9**	0.00	6	**424.9**	0.00	4	**424.9**	0.00	7
Gaskell67-22x5	**585.1**	**585.1**	0.00	9	**585.1**	0.00	6	**585.1**	0.00	11
Gaskell67-29x5	**512.1**	**512.1**	0.00	11	**512.1**	0.00	7	**512.1**	0.00	13
Gaskell67-32x5	**562.2**	**562.2**	0.00	40	**562.2**	0.00	20	**562.2**	0.00	43
Gaskell67-32x5	**504.3**	**504.3**	0.00	22	**504.3**	0.00	15	**504.3**	0.00	24
Gaskell67-36x5	**460.4**	**460.4**	0.00	39	**460.4**	0.00	22	**460.4**	0.00	42
Min92-27x5	**3062.0**	**3062.0**	0.00	11	**3062.0**	0.00	7	**3062.0**	0.00	13
Min92-134x8	**5709.0**	5890.6	3.18	252	5789.0	1.40	134	5920.8	3.71	226
Average		**4554.6**	**0.78**	**105**	**4547.1**	**0.66**	**53**	**4583.1**	**1.33**	**99**
NBKS		**7**			**7**			**7**		
NIBS		**0**			**0**			**0**		

Table 3.6: Summarized best results for all the algorithms on the complete data set

Set	Size	GRASP+ELS		SALRP		ALNS - 500K			ALNS - 5000K		GRASP+ILP			MACO		GTVNS	
		Gap Best PBKS	CPU time	Gap PBKS	CPU time	Gap Best PBKS	Gap Avg. PBKS	CPU time	Gap Best PBKS	CPU time	Gap Best PBKS	Gap Avg. PBKS	CPU time*	Gap Best PBKS	CPU time*	Gap PBKS	CPU time
Tuzun-Burke	36	1.15	607	1.35	826	0.29	0.75	830	0.04	8103	0.21	0.53	2255	1.09	202	0.62	201
Prodhon	30	1.08	258	0.43	422	0.41	0.69	451	0.23	4221	0.02	0.23	1130	0.36	191	0.34	91
Barreto	13	0.07	188	0.29	161	0.15	0.24	177	0.05	1772	0.13	0.62	241	0.06	49	0.66	53
Total	79																
Global Avg.		**0.95**	**405**	**0.82**	**564**	**0.31**	**0.64**	**579**	**0.11**	**5587**	**0.12**	**0.43**	**1496**	**0.65**	**173**	**0.52**	**135**
Total NBKS		**29**		**25**		**30**	**17**		**55**		**41**	**15**		**26**		**28**	
Total NIBS		**1**		**1**		**4**	**0**		**18**		**16**	**0**		**0**		**5**	
CPU		**Core 2 Quad (2.83 Ghz)**		**Core 2 Quad (2.66 Ghz)**		**AMD Opteron 275 (2.20 Ghz)**			**AMD Opteron 275 (2.20 Ghz)**		**Intel Xeon E5462 (3.00 Ghz)**			**Athlon XP 2500+ (1.83 Ghz)**		**Core 2 Duo (2.00 Ghz)**	
CPU index		**4373**		**4046**		**1234**			**1234**		**9586**			**374**		**1398**	

* For each instance: average CPU time over 10 runs

Table 3.7: Best results for all algorithms on Tuzun-Burke Instances

Instance	PBKS	GRASP+ELS Best Cost	GRASP+ELS Gap Best PBKS	GRASP+ELS CPU time	SALRP Cost	SALRP Gap PBKS	SALRP CPU time	ALNS - 500K Best Cost	ALNS - 500K Gap Best PBKS	ALNS - 500K Avg. Cost	ALNS - 500K Gap Avg. PBKS	ALNS - 500K CPU time	ALNS - 5000K Best Cost	ALNS - 5000K Gap Best PBKS	ALNS - 5000K CPU time	GRASP+ILP Best Cost	GRASP+ILP Gap Best PBKS	GRASP+ILP Avg. Cost	GRASP+ILP Gap Avg. PBKS	GRASP+ILP CPU time*	MACO Best Cost	MACO Gap Best PBKS	MACO CPU time*	GTVNS Cost	GTVNS Gap PBKS	GTVNS CPU time
111112	**1467.68**	1473.36	0.39	233	1477.24	0.65	369	**1467.68**	0.00	1475.67	0.54	275	**1467.68**	0.00	-	1468.20	0.04	1475.40	0.53	172	1489.68	1.50	71	1479.21	0.79	84
111122	**1449.20**	**1449.20**	0.00	9	1470.96	1.50	274	1452.14	0.20	1464.72	1.07	321	**1449.20**	0.00	-	**1449.20**	0.00	1454.20	0.35	474	1453.89	0.32	46	1485.28	2.49	126
111212	**1394.80**	1396.59	0.13	112	1408.65	0.99	231	1394.93	0.01	1400.49	0.41	244	**1394.80**	0.00	-	1396.60	0.13	1405.00	0.73	162	1407.78	0.93	61	1402.59	0.56	74
111222	**1432.29**	**1432.29**	0.00	114	**1432.29**	0.00	420	1433.42	0.08	1441.21	0.62	376	**1432.29**	0.00	-	1432.90	0.04	1445.40	0.92	506	1433.42	0.08	54	1463.23	2.16	99
112112	**1167.16**	**1167.16**	0.00	27	1177.14	0.86	348	1167.53	0.03	1173.04	0.50	489	**1167.16**	0.00	-	1176.30	0.78	1178.30	0.95	225	1208.04	3.50	80	**1167.16**	0.00	83
112122	**1102.24**	**1102.24**	0.00	259	1110.36	0.74	342	**1102.24**	0.00	1102.34	0.01	373	**1102.24**	0.00	-	1102.80	0.05	1106.00	0.34	415	**1102.24**	0.00	65	**1102.24**	0.00	105
112212	**791.66**	792.03	0.05	5	**791.66**	0.00	360	**791.66**	0.00	791.83	0.02	739	**791.66**	0.00	-	791.90	0.03	796.90	0.66	197	792.90	0.16	95	**791.66**	0.00	96
112222	**728.30**	**728.30**	0.00	48	731.95	0.50	418	**728.30**	0.00	728.32	0.00	384	**728.30**	0.00	-	**728.30**	0.00	728.40	0.01	371	**728.30**	0.00	65	**728.30**	0.00	126
113112	**1238.49**	1240.39	0.15	55	**1238.49**	0.00	300	1238.70	0.02	1240.31	0.15	357	**1238.49**	0.00	-	1239.40	0.07	1241.90	0.28	224	1265.27	2.16	77	**1238.49**	0.00	82
113122	**1245.31**	1246.00	0.06	233	1247.28	0.16	428	1246.52	0.10	1248.17	0.23	445	**1245.31**	0.00	-	1245.50	0.02	1246.40	0.09	472	1256.95	0.93	50	1247.27	0.16	127
113212	**902.26**	902.30	0.00	249	**902.26**	0.00	291	**902.26**	0.00	902.27	0.00	321	**902.26**	0.00	-	902.30	0.00	902.50	0.03	177	**902.26**	0.00	61	**902.26**	0.00	71
113222	**1018.29**	**1018.29**	0.00	196	1024.02	0.56	316	**1018.29**	0.00	1018.56	0.03	386	**1018.29**	0.00	-	**1018.29**	0.00	1019.60	0.13	496	**1018.29**	0.00	69	**1018.29**	0.00	85
131112	**1914.41**	1944.57	1.58	518	1953.85	2.06	743	1922.70	0.43	1939.52	1.31	504	**1914.41**	0.00	-	1928.00	0.71	1934.70	1.06	1073	1945.43	1.62	227	1933.67	1.01	179
131122	**1823.20**	1864.24	2.25	705	1899.05	4.16	835	1847.93	1.36	1857.29	1.87	635	1823.53	0.02	-	**1823.20**	0.00	1834.20	0.60	2021	1853.22	1.65	101	1852.14	1.59	173
131212	**1969.80**	1992.41	1.15	727	2057.53	4.45	456	1975.83	0.31	2009.44	2.01	664	1975.83	0.31	-	**1969.80**	0.00	1978.20	0.43	782	1991.44	1.10	201	1983.09	0.67	184
131222	**1792.80**	1835.25	2.37	415	1801.39	0.48	833	1806.31	0.75	1838.51	2.55	485	1796.45	0.20	-	**1792.80**	0.00	1800.20	0.41	1647	1812.34	1.09	141	1803.01	0.57	175
132112	**1444.73**	1453.78	0.63	103	1453.30	0.59	750	1447.43	0.19	1449.15	0.31	1049	1444.73	0.00	-	1447.50	0.19	1452.50	0.54	757	1499.05	3.76	206	**1443.32**	-0.10	186
132122	**1434.63**	1444.17	0.66	662	1455.50	1.45	828	1445.32	0.75	1446.91	0.86	805	**1434.63**	0.00	-	1443.80	0.64	1448.10	0.94	2863	1446.63	0.84	163	1441.43	0.47	210
132212	**1204.42**	1219.86	1.28	459	1206.24	0.15	752	1204.98	0.05	1205.83	0.12	2197	**1204.42**	0.00	-	1204.90	0.04	1206.10	0.14	959	1204.76	0.03	218	**1204.42**	0.00	128
132222	**931.28**	945.81	1.56	224	934.62	0.36	842	931.49	0.02	933.14	0.20	982	**931.28**	0.00	-	931.70	0.05	932.30	0.11	2466	931.73	0.05	150	**931.28**	0.00	177
133112	**1694.18**	1712.11	1.06	271	1720.81	1.57	742	1694.64	0.03	1700.39	0.37	1046	**1694.18**	0.00	-	1700.30	0.36	1711.70	1.03	992	1724.02	1.76	226	1701.34	0.42	182
133122	**1392.01**	1402.94	0.79	524	1415.85	1.71	833	1400.50	0.61	1403.50	0.83	925	**1392.01**	0.00	-	1400.10	0.58	1401.70	0.70	2016	1401.05	0.65	123	1416.74	1.78	175
133212	**1198.20**	1214.82	1.39	251	1216.84	1.56	756	1198.67	0.04	1199.27	0.09	1375	1198.28	0.01	-	**1198.20**	0.00	1200.50	0.19	895	1217.29	1.59	241	1213.87	1.31	207
133222	**1151.80**	1155.96	0.36	375	1159.12	0.64	837	1152.01	0.02	1154.36	0.22	911	**1151.80**	0.00	-	1157.70	0.51	1159.00	0.63	2640	1158.03	0.54	130	**1151.80**	0.00	208
121112	**2249.00**	2295.90	2.09	655	2324.10	3.34	1328	2265.15	0.72	2278.27	1.30	944	2251.93	0.13	-	**2249.00**	0.00	2258.80	0.44	2094	2304.67	2.48	461	2258.02	0.40	315
121122	**2153.80**	2203.57	2.31	432	2258.16	4.85	1455	2183.05	1.36	2192.61	1.80	847	2159.93	0.28	-	**2153.80**	0.00	2161.40	0.35	4911	2187.65	1.57	231	2166.20	0.58	300
121212	**2212.40**	2246.39	1.54	1566	2260.30	2.17	1319	2233.55	0.96	2247.75	1.60	907	2220.01	0.34	-	**2212.40**	0.00	2223.90	0.52	2304	2231.46	0.86	428	2239.65	1.23	287
121222	**2230.94**	2265.53	1.55	2192	2326.53	4.28	1428	**2230.94**	0.00	2263.20	1.45	860	**2230.94**	0.00	-	2232.50	0.07	2238.60	0.34	5176	2275.70	2.01	234	2236.73	0.26	351
122112	**2073.73**	2106.47	1.58	1521	2112.65	1.88	1320	2082.60	0.43	2093.78	0.97	1606	**2073.73**	0.00	-	2085.00	0.54	2094.50	1.00	3520	2098.56	1.20	570	2103.82	1.45	278
122122	**1692.17**	1779.05	5.13	618	1722.99	1.82	1400	1710.67	1.09	1732.00	2.35	941	**1692.17**	0.00	-	1703.80	0.69	1709.00	0.99	7178	1711.25	1.13	277	1717.92	1.52	433
122212	**1453.18**	1474.25	1.45	514	1469.10	1.10	1299	1458.55	0.37	1462.15	0.62	1861	**1453.18**	0.00	-	1465.90	0.88	1469.20	1.10	4163	1472.93	1.36	544	1469.45	1.12	318
122222	**1082.74**	1085.69	0.27	1243	1088.64	0.54	1429	1085.29	0.24	1086.08	0.31	812	1082.74	0.00	-	1083.90	0.11	1087.20	0.41	7194	1087.57	0.45	317	**1082.46**	-0.03	349
123112	**1960.30**	2004.33	2.25	1451	1994.16	1.73	1318	1964.75	0.23	1971.01	0.55	968	**1960.30**	0.00	-	1966.70	0.33	1971.70	0.58	3061	1978.74	0.94	387	1969.38	0.46	261
123122	**1926.64**	1964.40	1.96	1273	1932.05	0.28	1412	**1926.64**	0.00	1952.31	1.33	740	**1926.64**	0.00	-	1932.70	0.31	1941.60	0.78	9341	1959.71	1.72	230	1935.74	0.47	344
123212	**1762.03**	1778.80	0.95	1398	1779.10	0.97	1314	1762.09	0.00	1764.16	0.12	2055	**1762.03**	0.00	-	1765.80	0.21	1769.80	0.44	3814	1782.94	1.19	406	1776.90	0.84	349
123222	**1391.68**	1453.82	4.47	2202	1396.42	0.34	1427	1393.06	0.10	1395.38	0.27	1038	1391.68	0.00	-	1392.40	0.05	1393.90	0.16	5422	1392.70	0.07	269	**1391.50**	-0.01	317
Average		**1522.01**	**1.15**	**607**	**1526.41**	**1.35**	**826**	**1507.44**	**0.29**	**1515.64**	**0.75**	**830**	**1502.90**	**0.04**	**8103**	**1505.38**	**0.21**	**1510.52**	**0.53**	**2255**	**1520.22**	**1.09**	**202**	**1512.50**	**0.62**	**201**
NBKS		**6**			**4**			**8**		**0**			**26**			**10**		**0**			**4**			**13**		
NIBS		**0**			**0**			**3**		**0**			**14**			**7**		**0**			**0**			**3**		

* For each instance: average CPU time over 10 runs

Table 3.8: Best results for all algorithms on Prodhon Instances

Instance	PBKS	GRASP+ELS			SALRP			ALNS - 500K					ALNS - 5000K			GRASP+ILP					MACO			GTVNS		
		Best Cost	Gap Best PBKS	CPU time	Cost	Gap PBKS	CPU time	Best Cost	Gap Best PBKS	Avg. Cost	Gap Avg. PBKS	CPU time	Best Cost	Gap Best PBKS	CPU time	Best Cost	Gap Best PBKS	Avg. Cost	Gap Avg. PBKS	CPU time*	Best Cost	Gap Best PBKS	CPU time*	Cost	Gap PBKS	CPU time
20-5-1a	54793	54793	0.00	0	54793	0.00	20	54793	0.00	54793	0.00	39	54793	0.00	-	54793	0.00	54793	0.00	1	54793	0.00	4	54793	0.00	2
20-5-1b	39104	39104	0.00	0	39104	0.00	15	39104	0.00	39104	0.00	54	39104	0.00	-	39104	0.00	39104	0.00	2	39104	0.00	5	39104	0.00	3
20-5-2a	48908	48908	0.00	0	48908	0.00	19	48908	0.00	48908	0.00	38	48908	0.00	-	48908	0.00	48908	0.00	1	48908	0.00	4	48945	0.08	2
20-5-2b	37542	37542	0.00	0	37542	0.00	15	37542	0.00	37542	0.00	67	37542	0.00	-	37542	0.00	37542	0.00	2	37542	0.00	5	37542	0.00	3
50-5-1a	90111	90111	0.00	3	90111	0.00	75	90111	0.00	90111	0.00	101	90111	0.00	-	90111	0.00	90111	0.00	17	90111	0.00	25	90111	0.00	13
50-5-1b	63242	63242	0.00	0	63242	0.00	58	63242	0.00	63242	0.00	65	63242	0.00	-	63242	0.00	63248	0.01	17	63242	0.00	21	63242	0.00	9
50-5-2a	88298	88643	0.39	11	88298	0.00	95	88443	0.16	88576	0.31	99	88298	0.00	-	88298	0.00	88332	0.04	15	88298	0.00	24	89342	1.18	12
50-5-2b	67308	67308	0.00	16	67340	0.05	59	67340	0.05	67448	0.21	200	67308	0.00	-	67373	0.10	67554	0.37	19	67308	0.00	20	67951	0.96	10
50-5-2bis	84055	84055	0.00	0	84055	0.00	75	84055	0.00	84119	0.08	107	84055	0.00	-	84055	0.00	84055	0.00	18	84055	0.00	25	84126	0.08	8
50-5-2bbis	51822	51822	0.00	11	51822	0.00	66	51822	0.00	51840	0.03	98	51822	0.00	-	51883	0.12	51898	0.15	24	51822	0.00	17	52213	0.75	9
50-5-3a	86203	86203	0.00	0	86456	0.29	74	86203	0.00	86262	0.07	101	86203	0.00	-	86203	0.00	86203	0.00	15	86203	0.00	33	86203	0.00	18
50-5-3b	61830	61830	0.00	0	62700	1.41	58	61830	0.00	61830	0.00	137	61830	0.00	-	61830	0.00	61830	0.00	20	61830	0.00	26	61885	0.09	20
100-5-1a	275406	276960	0.56	148	277035	0.59	349	275636	0.08	276364	0.35	520	275524	0.04	-	275406	0.00	275626	0.08	189	276220	0.30	117	276137	0.27	75
100-5-1b	213704	215854	1.01	68	216002	1.08	269	214735	0.48	215059	0.63	1190	213704	0.00	-	214308	0.28	214699	0.47	179	214323	0.29	135	216154	1.15	59
100-5-2a	193671	194267	0.31	212	194124	0.23	349	193752	0.04	193903	0.12	463	193671	0.00	-	193769	0.05	194118	0.23	107	194441	0.40	238	193896	0.12	76
100-5-2b	157095	157375	0.18	125	157150	0.04	212	157095	0.00	157157	0.04	859	157095	0.00	-	157157	0.04	157238	0.09	95	157222	0.08	144	157180	0.05	82
100-5-3a	200242	200345	0.05	141	200242	0.00	250	200305	0.03	200496	0.13	454	200246	0.00	-	200277	0.02	200341	0.05	87	201038	0.40	179	200777	0.27	69
100-5-3b	152441	152528	0.06	221	152467	0.02	197	152441	0.00	152900	0.30	684	152441	0.00	-	152441	0.00	152737	0.19	96	152722	0.18	152	153435	0.65	68
100-10-1a	288415	301418	4.51	48	291043	0.91	270	296877	2.93	299982	4.01	210	292868	1.54	-	288415	0.00	293117	1.63	1841	291134	0.94	105	287864	-0.19	203
100-10-1b	230989	269594	16.71	186	234210	1.39	203	235849	2.10	240829	4.26	188	233146	0.93	-	230989	0.00	233416	1.05	2330	235348	1.89	82	232599	0.70	117
100-10-2a	243695	243778	0.03	260	245813	0.87	261	244740	0.43	245548	0.76	136	243829	0.05	-	243695	0.00	244022	0.13	212	245263	0.64	123	245484	0.73	52
100-10-2b	203988	203988	0.00	139	205312	0.65	199	204016	0.01	204494	0.25	261	203988	0.00	-	203988	0.00	204200	0.10	243	205524	0.75	85	204252	0.13	42
100-10-3a	250882	253511	1.05	164	250882	0.00	338	253801	1.16	254882	1.59	202	253722	1.13	-	250882	0.00	252371	0.59	2576	254302	1.36	113	254558	1.47	82
100-10-3b	204601	205087	0.24	203	205009	0.20	240	205609	0.49	206175	0.77	224	204601	0.00	-	204602	0.00	204996	0.19	1006	204786	0.09	79	205824	0.60	78
200-10-1a	475344	486467	2.34	1521	481002	1.19	1428	480883	1.17	483205	1.65	752	478951	0.76	-	475344	0.00	476674	0.28	3786	478843	0.74	942	477009	0.35	320
200-10-1b	377043	382329	1.40	359	383586	1.74	1336	378961	0.51	380538	0.93	1346	378065	0.27	-	377043	0.00	378781	0.46	3647	378351	0.35	562	377716	0.18	239
200-10-2a	449152	452276	0.70	112	450848	0.38	1796	450451	0.29	451750	0.58	1201	450377	0.27	-	449152	0.00	449469	0.07	5216	451457	0.51	704	449006	-0.03	231
200-10-2b	374469	376027	0.42	1610	376674	0.59	1245	374751	0.08	376112	0.44	1349	374751	0.08	-	374469	0.00	375053	0.16	2832	374972	0.13	404	374717	0.07	290
200-10-3a	469706	478380	1.85	1596	473875	0.89	1776	475373	1.21	479366	2.06	1251	474087	0.93	-	469706	0.00	471218	0.32	4357	475155	1.16	879	471978	0.48	330
200-10-3b	362743	365166	0.67	591	363701	0.26	1326	366902	1.15	366902	1.15	1137	366416	1.01	-	362743	0.00	363755	0.28	4937	365401	0.73	491	362827	0.02	214
Average		199630	1.08	258	197778	0.43	422	197852	0.41	198648	0.69	451	197357	0.23	4221	196591	0.02	197180	0.23	1130	197657	0.36	191	197229	0.34	91
NBKS		12			11			12		7			18			21		8			12			8		
NIBS		0			1			1		0			4			8		0			0			2		

* For each instance: average CPU time over 10 runs

Table 3.9: Best results for all algorithms on Barreto Instances

Instance	PBKS	GRASP+ELS			SALRP			ALNS - 500K					ALNS - 5000K			GRASP+ILP					MACO			GTVNS		
		Best Cost	Gap Best PBKS	CPU time	Cost	Gap PBKS	CPU time	Best Cost	Gap Best PBKS	Avg. Cost	Gap Avg. PBKS	CPU time	Best Cost	Gap Best PBKS	CPU time	Best Cost	Gap Best PBKS	Avg. Cost	Gap Avg. PBKS	CPU time*	Best Cost	Gap Best PBKS	CPU time*	Cost	Gap PBKS	CPU time
Christofides69-50x5	**565.6**	**565.6**	0.00	8	**565.6**	0.00	53	**565.6**	0.00	**565.6**	0.00	73	**565.6**	0.00	-	**565.6**	0.00	581.0	2.72	15	**565.6**	0.00	29	580.4	2.62	22
Christofides69-75x10	**844.4**	850.8	0.76	86	848.9	0.53	127	853.5	1.08	854.9	1.24	207	848.9	0.53	-	**844.4**	0.00	848.3	0.46	74	844.9	0.06	59	853.8	1.11	45
Christofides69-100x10	**833.4**	**833.4**	0.00	127	838.3	0.59	331	**833.4**	0.00	835.4	0.24	403	**833.4**	0.00	-	841.7	1.00	851.0	2.11	351	836.8	0.40	84	837.1	0.44	111
Daskin95-88x8	**355.8**	**355.8**	0.00	130	**355.8**	0.00	577	**355.8**	0.00	**355.8**	0.00	250	**355.8**	0.00	-	**355.8**	0.00	356.1	0.08	164	**355.8**	0.00	100	361.6	1.63	97
Daskin95-150x10	**43963.6**	**43963.6**	0.00	1697	45109.4	2.61	323	44309.0	0.79	44497.2	1.21	613	44004.9	0.09	-	44179.0	0.49	44321.3	0.81	1311	44131.0	0.38	167	44578.9	1.40	199
Gaskell67-21x5	**424.9**	**424.9**	0.00	0	**424.9**	0.00	18	**424.9**	0.00	**424.9**	0.00	25	**424.9**	0.00	-	**424.9**	0.00	**424.9**	0.00	1	**424.9**	0.00	6	**424.9**	0.00	4
Gaskell67-22x5	**585.1**	**585.1**	0.00	15	**585.1**	0.00	17	**585.1**	0.00	**585.1**	0.00	21	**585.1**	0.00	-	**585.1**	0.00	**585.1**	0.00	3	**585.1**	0.00	5	**585.1**	0.00	6
Gaskell67-29x5	**512.1**	**512.1**	0.00	9	**512.1**	0.00	24	**512.1**	0.00	**512.1**	0.00	40	**512.1**	0.00	-	**512.1**	0.00	**512.1**	0.00	5	**512.1**	0.00	9	**512.1**	0.00	7
Gaskell67-32x5	**562.2**	**562.2**	0.00	18	**562.2**	0.00	27	**562.2**	0.00	**562.2**	0.00	58	**562.2**	0.00	-	**562.2**	0.00	**562.2**	0.00	5	**562.2**	0.00	13	**562.2**	0.00	20
Gaskell67-32x5	**504.3**	**504.3**	0.00	34	**504.3**	0.00	25	**504.3**	0.00	**504.3**	0.00	55	**504.3**	0.00	-	**504.3**	0.00	**504.3**	0.00	6	**504.3**	0.00	10	**504.3**	0.00	15
Gaskell67-36x5	**460.4**	**460.4**	0.00	0	**460.4**	0.00	32	**460.4**	0.00	**460.4**	0.00	61	**460.4**	0.00	-	**460.4**	0.00	**460.4**	0.00	7	**460.4**	0.00	13	**460.4**	0.00	22
Min92-27x5	**3062.0**	**3062.0**	0.00	35	**3062.0**	0.00	23	**3062.0**	0.00	**3062.0**	0.00	38	**3062.0**	0.00	-	**3062.0**	0.00	**3062.0**	0.00	3	**3062.0**	0.00	9	**3062.0**	0.00	7
Min92-134x8	**5709.0**	5719.3	0.18	280	**5709.0**	0.00	522	5713.0	0.07	5732.6	0.41	460	**5709.0**	0.00	-	5719.3	0.18	5816.7	1.89	1189	**5709.0**	0.00	137	5789.0	1.40	134
Average		**4492.27**	**0.07**	**188**	**4579.85**	**0.29**	**161**	**4518.56**	**0.15**	**4534.81**	**0.24**	**177**	**4494.51**	**0.05**	**1772**	**4508.98**	**0.13**	**4529.64**	**0.62**	**241**	**4504.16**	**0.06**	**49**	**4547.06**	**0.66**	**53**
NBKS		**11**			**10**			**10**		**9**			**11**			**10**		**7**			**10**			**7**		
NIBS		**1**			**0**			**0**		**0**			**0**			**1**		**0**			**0**			**0**		

* For each instance: average CPU time over 10 runs

Chapter 4

A heuristic algorithm for the MDVRP

Notes about the chapter

The contents of this chapter is based on the paper entitled "*A Hybrid Granular Tabu Search algorithm for the Multi-Depot Vehicle Routing Problem*", co-authored with Rodrigo Linfati, Professor Maria Gulnara Baldoquin and Professor Paolo Toth, which has been submitted for publication. Partial results will be presented in the conference TRISTAN VII, San Pedro Atacama-Chile (2013).

4.1 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 [60]. 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.

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 [60]. The second strategy is based on a penalty approach proposed by Gendreau et al. [21] and Taillard [57]. 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. [25], 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 procedure, and (2) the Granular Tabu Search procedure. Algorithm ELTG is based on the heuristic framework proposed by Escobar et al. [19] for the *Capacitated Location Routing Problem* (CLRP). The main differences of algorithm ELTG with respect to the algorithm presented in Escobar et al. [19] are: i) the hybrid procedure used for the construction of the initial solution, ii) the penalty diversification strategy, and iii) the new local search procedures proposed within the main loop of the Granular Tabu Search phase.

4.2 Initial Solution

The initial MDVRP solution S_0 is constructed by using a hybrid heuristic 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* (LKH) (for further details see Lin and Kernighan [35] and Helsgaun [28]).

- *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 the 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).

- *Step 3.* For each depot i and each cluster g, a TSP tour is determined, by using procedure LKH, to obtain the traveling cost (l_{ig}) between depot i and the customers belonging to cluster g.

- *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 :

$$min\, z = \sum_{i\in I}\sum_{g\in G} l_{ig}x_{ig} + \sigma \sum_{i\in I}\sum_{g\in G} max(0, \bar{d_{ig}} - D)x_{ig} \tag{4.1}$$

subject to

$$\sum_{i\in I} x_{ig} = 1 \qquad \forall g \in G \tag{4.2}$$

$$\sum_{j\in G} x_{ig} \leq k \qquad \forall i \in I \tag{4.3}$$

$$x_{ig} \in \{0,1\} \qquad \forall i \in I, g \in G \tag{4.4}$$

where:
I *set of depots*
G *set of clusters*
σ *penalty factor*
$\bar{d_{ig}} = l_{ig} + \sum_{j\in G}\delta_j$ *where* $\bar{d_{ig}}$ *is duration of cluster* $g \in G$ *when* g *is assigned to the depot* $i \in I$

The objective function (4.1) 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 (4.2) guarantee that each cluster is assigned to exactly one depot. Constraints (4.3) guarantee that the number of clusters assigned to each depot must not exceed the number k of vehicles available at each depot.

Constraints (4.4) can be replaced by $x_{ig} \geq 0, \forall i \in I, \forall g \in G$, and model (4.1) - (4.4) 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 [27]) 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 to 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. [19] 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.

4.3 Granular Tabu Search

Algorithm ELTG allows solutions which are infeasible with respect to the vehicle capacities and the duration of the routes (see Subsection 4.3.2). 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 Subsection 4.3.2, 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.

4.3.1 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 [35] 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 [63]). 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 Section 4.3.2) is performed, even if solution S is worse than the current solution.

4.3.2 Search, Intensification and Diversification strategies

The proposed algorithm, as in that presented in Gendreau et al. [21], 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 γ_{min}, γ_{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 [60]. 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 [57]. 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. [21], 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).

4.3.3 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).

4.4 Computational experiments

4.4.1 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 (4.1) - (4.4) 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 [10]. Instances 8-11 have been described in Gillett and Johnson [22]. Instances 12-23 were proposed by Chao et al. [9]. Finally, instances 24-33 were introduced by \Cordeau et al. [15]. 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 3.2) with the most effective published heuristic algorithms proposed for the MDVRP: Tabu Search (CGL97) of Cordeau et al. [15], the general heuristic (PR07) of Pisinger and Ropke [44], the hybrid genetic algorithm (VCGLR12) of Vidal et al. [62], and the sequential tabu search algorithm (CM12) of Cordeau and Maischberger [14].

For each instance, only one run of algorithm ELTG is executed. 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. [21]) to a uniformly distributed random integer number in the interval $[5, 10]$. 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 3.1 and 3.2, for each instance, the following notation is used:

Instance	instance number;
n	number of customers;
m	number of depots;
k	maximum number of available vehicles at each depot;
D	maximum duration of each route;
Q	capacity of each vehicle;
Cost	solution cost obtained by the corresponding algorithm;
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;
Gap BKS	percentage gap of the solution cost found by the

	corresponding algorithm with respect to the value of BKS;
Status	status of solutions obtained by the initial hybrid procedure (*feasible* or *infeasible*);
Time	running time in seconds on 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. [9] (CGW93) and by Renaud et al. [52] (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 [3]. 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 [1]). 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).

4.4.2 Global results

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

- Initial Solution: solution obtained after the application of the Initial Hybrid procedure;
- Granular Tabu Search: solution obtained by the proposed heuristic ELTG (i.e. at the end of the Granular Tabu Search procedure).

Whenever a solution obtained by the initial hybrid procedure is infeasible with respect 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.

A summary on the results obtained by the five considered algorithms (CGL97, PR07, VCGLR12, CM12, and ELTG) for the complete set of instances is given in Table 3.2. In this table we report the results as presented in the corresponding papers.

Algorithms PR07 and VCGLR12 have been executed for ten runs. The results reported for both algorithms correspond, for each instance, to the average cost found and to the average CPU time over the ten runs. For algorithm CM12, the results reported correspond, for each instance, to the average cost found and to the average CPU time obtained over 10 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.

Table 3.2 shows that algorithm ELTG provides the lowest global average value of Gap BKS on the first 23 instances. For instances 24 - 33, algorithm ELTG has a global average value of Gap BKS smaller than that of algorithms CGL97, PR07, and CM12; only algorithm VCGLR12 provides, although with longer CPU times, a better global average value of Gap BKS. 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, algorithm ELTG obtains the best results. Indeed, the proposed algorithm is able to find, within short CPU times, 20 best known solutions, and to improve the previous best known solution for 3 instances.

As for the average CPU time, algorithm ELTG is faster than algorithms VCGLR12 and CM12, which were able to find the previous best results in terms of the average value of Gap BKS and of the values of NBKS and NIBS. On the other hand, the average running time of algorithm ELTG is larger than that of algorithms CGL97 and PR07. This can be explained by considering that algorithm ELTG uses several improvement procedures in the main loop of the Granular Tabu Search phase. Although the average running time of algorithm ELTG is larger than that of these two approaches, it remains within acceptable values for an operational problem like the MDVRP.

4.5 Concluding remarks

We propose an effective Hybrid Granular Tabu Search algorithm for the Multi Depot Vehicle Routing Problem (MDVRP). 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 Vehicle Routing Problem with Heterogeneous Fleet (HMDVRP), and other problems obtained by adding constraints as time windows, pickups and deliveries, etc.

Table 4.1: Solutions obtained by each phase of the proposed algorithm

Characteristics of Instances						BKS	Initial Solution				Granular Tabu Search		
Instance	n	m	k	D	Q		Cost	Gap BKS	Time	Status	Cost	Gap BKS	Time
1	50	4	4	∞	80	**576.87**	594.52	3.06	5	Feasible	**576.87**	0.00	7
2	50	4	2	∞	160	**473.53**	492.18	3.94	4	Feasible	**473.53**	0.00	6
3	75	5	3	∞	140	**641.19**	695.37	8.45	19	Feasible	**641.19**	0.00	29
4	100	2	8	∞	100	**1001.04**	1018.47	1.74	62	Feasible	**1001.04**	0.00	90
5	100	2	5	∞	200	**750.03**	751.26	0.16	17	Feasible	**750.03**	0.00	26
6	100	3	6	∞	100	**876.50**	918.29	4.77	65	Feasible	**876.50**	0.00	103
7	100	4	4	∞	100	**881.97**	945.00	7.15	76	Feasible	884.66	0.31	106
8	249	2	14	310	500	**4372.78**	4584.97	4.85	185	Feasible	**4371.66**	-0.03	285
9	249	3	12	310	500	**3858.66**	4009.69	3.91	156	Feasible	3880.85	0.58	256
10	249	4	8	310	500	**3631.11**	3854.68	6.16	166	Feasible	**3629.60**	-0.04	267
11	249	5	6	310	500	**3546.06**	3738.33	5.42	125	Feasible	**3545.18**	-0.02	192
12	80	2	5	∞	60	**1318.95**	1369.47	3.83	4	Feasible	**1318.95**	0.00	6
13	80	2	5	200	60	**1318.95**	1349.07	2.28	5	Feasible	**1318.95**	0.00	7
14	80	2	5	180	60	**1360.12**	**1360.12**	0.00	4	Feasible	**1360.12**	0.00	6
15	160	4	5	∞	60	**2505.42**	2590.87	3.41	69	Feasible	**2505.42**	0.00	114
16	160	4	5	200	60	**2572.23**	2761.25	7.35	87	Feasible	**2572.23**	0.00	118
17	160	4	5	180	60	**2709.09**	2895.76	6.89	79	Feasible	**2709.09**	0.00	108
18	240	6	5	∞	60	**3702.85**	4111.78	11.04	178	Feasible	**3702.85**	0.00	278
19	240	6	5	200	60	**3827.06**	4292.11	12.15	176	Feasible	**3827.06**	0.00	256
20	240	6	5	180	60	**4058.07**	4441.59	9.45	190	Infeasible	**4058.07**	0.00	267
21	360	9	5	∞	60	**5474.84**	6106.37	11.54	166	Feasible	**5474.84**	0.00	268
22	360	9	5	200	60	**5702.16**	6613.80	15.99	170	Infeasible	**5702.16**	0.00	262
23	360	9	5	180	60	**6078.75**	6677.53	9.85	199	Infeasible	6095.46	0.27	285
Avg.								**6.23**	**96**			**0.05**	**145**
24	48	4	1	500	200	**861.32**	894.26	3.82	2	Feasible	**861.32**	0.00	4
25	96	4	2	480	195	**1307.34**	1449.20	10.85	8	Infeasible	1311.11	0.29	11
26	144	4	3	460	190	**1803.80**	1883.80	4.44	72	Feasible	**1803.80**	0.00	118
27	192	4	4	440	185	**2058.31**	2103.46	2.19	89	Feasible	2064.11	0.28	124
28	240	4	5	420	180	**2331.20**	2466.38	5.80	147	Feasible	2349.63	0.79	213
29	288	4	6	400	175	**2676.30**	2769.73	3.49	145	Feasible	2710.30	1.27	234
30	72	6	1	500	200	**1089.56**	1255.87	15.26	8	Feasible	**1089.56**	0.00	11
31	144	6	2	475	190	**1664.85**	1883.39	13.13	47	Feasible	1665.50	0.04	66
32	216	6	3	450	180	**2133.20**	2258.09	5.85	94	Feasible	2151.45	0.86	156
33	288	6	4	425	170	**2868.26**	3046.00	6.20	199	Feasible	2910.78	1.48	302
Avg.								**7.10**	**81**			**0.50**	**124**
G. Avg								**6.50**	**91**			**0.18**	**139**

Table 4.2: Solutions (CPU Times) obtained by the MDVRP Algorithms

Characteristics of the Instances						Previous Solutions		CGL97 (1 run)			PR07 (Avg. 10 runs)			VCGLR12 (Avg. 10 runs)			CM12 (Avg. 10 runs)			ELTG (1 run)		
Instance	**n**	**m**	**k**	**D**	**Q**	**BKS**	**Ref. BKS**	**Cost**	**Gap BKS**	**Time**	**Cost**	**Gap BKS**	**Time**	**Cost**	**Gap BKS**	**Time**	**Cost**	**Gap BKS**	**Time**	**Cost**	**Gap BKS**	**Time**
1	50	4	4	∞	80	**576.87**	CGW93	**576.87**	0.00	194	**576.87**	0.00	29	**576.87**	0.00	14	**576.87**	0.00	-	**576.87**	0.00	7
2	50	4	2	∞	160	**473.53**	RLB96	473.87	0.07	208	**473.53**	0.00	28	**473.53**	0.00	13	**473.53**	0.00	-	**473.53**	0.00	6
3	75	5	3	∞	140	**641.19**	CGW93	645.15	0.62	340	**641.19**	0.00	64	**641.19**	0.00	26	**641.19**	0.00	-	**641.19**	0.00	29
4	100	2	8	∞	100	**1001.04**	PR07	1006.66	0.56	467	1006.09	0.50	88	1001.23	0.02	116	1002.64	0.16	-	**1001.04**	0.00	90
5	100	2	5	∞	200	**750.03**	CGL97	753.34	0.44	493	752.34	0.31	120	**750.03**	0.00	64	750.41	0.05	-	**750.03**	0.00	26
6	100	3	6	∞	100	**876.50**	RLB96	877.84	0.15	459	883.01	0.74	93	**876.50**	0.00	68	877.03	0.06	-	**876.50**	0.00	103
7	100	4	4	∞	100	**881.97**	PR07	891.95	1.13	463	889.36	0.84	88	884.43	0.28	93	884.18	0.25	-	884.66	0.31	106
8	249	2	14	310	500	**4372.78**	VCGLR12	4482.44	2.51	1526	4421.03	1.10	333	4397.42	0.56	600	4438.47	1.50	-	**4371.66**	-0.03	285
9	249	3	12	310	500	**3858.66**	VCGLR12	3920.85	1.61	1604	3892.50	0.88	361	3868.59	0.26	570	3894.10	0.92	-	3880.85	0.58	256
10	249	4	8	310	500	**3631.11**	VCGLR12	3714.65	2.30	1530	3666.85	0.98	363	3636.08	0.14	589	3660.39	0.81	-	**3629.60**	-0.04	267
11	249	5	6	310	500	**3546.06**	PR07	3580.84	0.98	1555	3573.23	0.77	357	3548.25	0.06	428	3553.88	0.22	-	**3545.18**	-0.02	192
12	80	2	5	∞	60	**1318.95**	RLB96	**1318.95**	0.00	334	1319.13	0.01	75	**1318.95**	0.00	31	**1318.95**	0.00	-	**1318.95**	0.00	6
13	80	2	5	200	60	**1318.95**	RLB96	**1318.95**	0.00	335	**1318.95**	0.00	60	**1318.95**	0.00	34	**1318.95**	0.00	-	**1318.95**	0.00	7
14	80	2	5	180	60	**1360.12**	CGL97	**1360.12**	0.00	326	**1360.12**	0.00	58	**1360.12**	0.00	33	**1360.12**	0.00	-	**1360.12**	0.00	6
15	160	4	5	∞	60	**2505.42**	CGL97	2534.13	1.15	844	2519.64	0.57	253	**2505.42**	0.00	115	**2505.42**	0.00	-	**2505.42**	0.00	114
16	160	4	5	200	60	**2572.23**	RLB96	**2572.23**	0.00	843	2573.95	0.07	188	**2572.23**	0.00	118	**2572.23**	0.00	-	**2572.23**	0.00	118
17	160	4	5	180	60	**2709.09**	CGL97	2720.23	0.41	822	**2709.09**	0.00	179	**2709.09**	0.00	128	**2709.09**	0.00	-	**2709.09**	0.00	108
18	240	6	5	∞	60	**3702.85**	CGL97	3710.49	0.21	1491	3736.53	0.91	419	**3702.85**	0.00	271	3703.96	0.03	-	**3702.85**	0.00	278
19	240	6	5	200	60	**3827.06**	RLB96	**3827.06**	0.00	1512	3838.76	0.31	315	**3827.06**	0.00	252	**3827.06**	0.00	-	**3827.06**	0.00	256
20	240	6	5	180	60	**4058.07**	CGL97	**4058.07**	0.00	1483	4064.76	0.16	300	**4058.07**	0.00	262	**4058.07**	0.00	-	**4058.07**	0.00	267
21	360	9	5	∞	60	**5474.84**	CGL97	5535.99	1.12	2890	5501.58	0.49	582	5476.41	0.03	600	5486.91	0.22	-	**5474.84**	0.00	268
22	360	9	5	200	60	**5702.16**	CGL97	5716.01	0.24	2934	5722.19	0.35	462	**5702.16**	0.00	600	5708.44	0.11	-	**5702.16**	0.00	262
23	360	9	5	180	60	**6078.75**	PR07	6139.73	1.00	2872	6092.66	0.23	443	**6078.75**	0.00	600	6086.05	0.12	-	6095.46	0.27	285
Avg.									**0.63**	**1110**		**0.40**	**229**		**0.06**	**245**		**0.19**	**-**		**0.05**	**145**
24	48	4	1	500	200	**861.32**	CGL97	**861.32**	0.00	242	**861.32**	0.00	30	**861.32**	0.00	10	**861.32**	0.00	-	**861.32**	0.00	4
25	96	4	2	480	195	**1307.34**	PR07	1314.99	0.59	505	1308.17	0.06	103	**1307.34**	0.00	46	1307.73	0.03	-	1311.11	0.29	11
26	144	4	3	460	190	**1803.80**	VCGLR12	1815.62	0.66	854	1810.66	0.38	214	**1803.80**	0.00	115	1805.24	0.08	-	**1803.80**	0.00	118
27	192	4	4	440	185	**2058.31**	VCGLR12	2094.24	1.75	1158	2073.16	0.72	296	2059.36	0.05	313	2071.48	0.64	-	2064.11	0.28	124
28	240	4	5	420	180	**2331.20**	VCGLR12	2408.10	3.30	1529	2350.31	0.82	372	2340.29	0.39	574	2355.44	1.04	-	2349.63	0.79	213
29	288	4	6	400	175	**2676.30**	VCGLR12	2768.13	3.43	2007	2695.74	0.73	465	2681.93	0.21	600	2699.58	0.87	-	2710.30	1.27	234
30	72	6	1	500	200	**1089.56**	CGL97	1092.12	0.24	412	**1089.56**	0.00	58	**1089.56**	0.00	20	**1089.56**	0.00	-	**1089.56**	0.00	11
31	144	6	2	475	190	**1664.85**	PR07	1676.26	0.69	906	1675.74	0.65	207	1665.05	0.01	123	1666.85	0.12	-	1665.50	0.04	66
32	216	6	3	450	180	**2133.20**	VCGLR12	2176.79	2.04	1462	2144.84	0.55	350	2134.17	0.05	366	2150.48	0.81	-	2151.45	0.86	156
33	288	6	4	425	170	**2868.26**	VCGLR12	3089.62	7.72	2105	2905.43	1.30	455	2886.59	0.64	600	2911.86	1.52	-	2910.78	1.48	302
Avg.									**2.04**	**1118**		**0.52**	**255**		**0.13**	**277**		**0.51**	**-**		**0.50**	**124**
G. Avg									**1.06**	**1112**		**0.44**	**237**		**0.08**	**254**		**0.29**	**1101**		**0.18**	**139**
NBKS								**8**			**8**			**20**			**13**			**23**		
NIBS								**0**			**0**			**2**			**0**			**5**		
CPU								**Sun Sparcstation 10**			**Pentium 4 (3.0 GHz)**			**AMD Opteron 250 (2.4 GHz)**			**Xeon X7350 (2.93 Ghz)**			**Core Duo (2.0 GHz)**		
CPU index								**---**			**489**			**1411**			**16715**			**1398**		

Bibliography

[1] PassMark Performance Test. `http://www.passmark.com`, 2012. [Online; accessed 28-Jun-2012].

[2] M. Albareda-Sambola, J.A. Díaz, and E. Fernández. A compact model and tight bounds for a combined location-routing problem. *Computers and Operations Research*, 32(3):407–428, 2005.

[3] R. Baldacci and A. Mingozzi. A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120(2):347–380, 2009.

[4] R. Baldacci, A. Mingozzi, and R.W. Calvo. An exact method for the capacitated location-routing problem. *Operations research*, 59(5):1284–1296, 2011.

[5] J. Barcelo and J. Casanovas. A heuristic lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research*, 15(2):212–226, 1984.

[6] S. Barreto, C. Ferreira, J. Paixao, and B.S. Santos. Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, 179(3):968–977, 2007.

[7] SS Barreto. Análise e modelização de problemas de localização-distribuição (analysis and modelling of location-routing problems). *PhD thesis, University of Aveiro*, pages 3810–4193, 2004.

[8] J.M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. Wolfler-Calvo. A branch-and-cut method for the capacitated location-routing problem. *Computers and Operations Research*, 38(6):931–941, 2011.

[9] I.M. Chao, B.L. Golden, and E. Wasil. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *American Journal of Mathematical and Management Sciences*, 13(3-4): 371–406, 1993.

[10] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly*, 20(3):309–318, 1969.

[11] G. Clarke and JW Wright. Scheduling of vehicles form a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[12] C. Contardo, J.F. Cordeau, and B. Gendron. A branch-and-cut-and-price algorithm for the capacitated location-routing problem. Technical report, Université de Montréal, 2011.

[13] C. Contardo, J.F. Cordeau, and B. Gendron. A grasp+ ilp-based metaheuristic for the capacitated location-routing problem. Technical report, Université de Montréal, 2011.

[14] J.F. Cordeau and M. Maischberger. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers and Operations Research*, 39(9):2033–2050, 2012.

[15] J.F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2): 105–19, 1997.

[16] C. Daganzo. *Logistics systems analysis*. Springer, 2005.

[17] G. Dueck. New optimization heuristics. *Journal of Computational Physics*, 104(1):86–92, 1993.

[18] C. Duhamel, P. Lacomme, C. Prins, and C. Prodhon. A graspxels approach for the capacitated location-routing problem. *Computers and Operations Research*, 37(11):1912–1923, 2010.

[19] J.W. Escobar, R. Linfati, and P. Toth. A two-phase hybrid metaheuristic algorithm for the capacitated location-routing problem. *Computers and Operations Research*, 2012. doi: 10.1016/j.cor.2012.05.008. URL `http://dx.doi.org/10.1016/j.cor.2012.05.008`.

[20] M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992.

[21] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

[22] B.E. Gillett and J.G. Johnson. Multi-terminal vehicle-dispatch algorithm. *Omega*, 4(6):711–718, 1976.

[23] B.E. Gillett and L.R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974.

[24] B.L. Golden, T.L. Magnanti, and H.Q. Nguyen. Implementing vehicle routing algorithms. *Networks*, 7(2):113–148, 1977.

[25] C. Groer, B. Golden, and E. Wasil. A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, 2(2):79–101, 2010.

[26] P.H. Hansen, B. Hegedahl, S. Hjortkjaer, and B. Obel. A heuristic solution to the warehouse location-routing problem. *European Journal of Operational Research*, 76(1):111–127, 1994.

[27] I. Heller and CB Tompkins. An extension of a theorem of dantzig. *Annals of Mathematics Studies*, 38:247–254, 1956.

[28] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1): 106–130, 2000.

[29] V.C. Hemmelmayr, J.F. Cordeau, and T.G. Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Technical report, Université de Montréal, 2011.

[30] J.G. Klincewicz and H. Luss. A lagrangian relaxation heuristic for capacitated facility location with single-source constraints. *Journal of the Operational Research Society*, 37(5):495–500, 1986.

[31] G. Laporte, Y. Nobert, and D. Arpin. Optimal solutions to capacitated multi-depot vehicle routing problems. *Congressus Numerantium*, 44: 283–292, 1984.

[32] G. Laporte, Y. Nobert, and D. Arpin. An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research*, 6 (9):291–310, 1986.

[33] G. Laporte, Y. Nobert, and S. Taillefer. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science*, 22(3):161–72, 1988.

[34] F. Li, B. Golden, and E. Wasil. Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers and Operations Research*, 32(5):1165–1179, 2005.

[35] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[36] J. Melechovskỳ, C. Prins, and R. Wolfler-Calvo. A metaheuristic to solve a location-routing problem with non-linear costs. *Journal of Heuristics*, 11(5):375–391, 2005.

[37] H. Min, V. Jayaraman, and R. Srivastava. Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research*, 108(1):1–15, 1998.

[38] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

[39] J.A. Moreno Pérez, J. Marcos Moreno-Vega, and I. Rodríguez Martín. Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research*, 151(2):365–378, 2003.

[40] G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2007.

[41] B. Ombuki-Berman and F. Hanshar. Using genetic algorithms for multi-depot vehicle routing. *Studies in Computational Intelligence*, 161:77–99, 2009.

[42] J. Perl and M.S. Daskin. A warehouse location-routing problem. *Transportation Research Part B: Methodological*, 19(5):381–396, 1985.

[43] S. Pirkwieser and G. Raidl. Variable neighborhood search coupled with ilp-based very large neighborhood searches for the (periodic) location-routing problem. *Hybrid Metaheuristics*, pages 174–189, 2010.

[44] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

[45] C. Prins, C. Prodhon, and R. Wolfler-Calvo. Nouveaux algorithmes pour le problême de localisation et routage sous contraintes de capacité. In *MOSIM (4éme Conférence Francophone de Modélisation et Simluation, Nantes, France*, volume 4, pages 1115–1122, 2004.

[46] C. Prins, C. Prodhon, and R. Wolfler-Calvo. A memetic algorithm with population management (ma| pm) for the capacitated location-routing problem. *Lecture Notes in Computer Science*, 3906:183–194, 2006.

[47] C. Prins, C. Prodhon, and R. Wolfler-Calvo. Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR: A Quarterly Journal of Operations Research*, 4(3):221–238, 2006.

[48] C. Prins, C. Prodhon, A. Ruiz, P. Soriano, and R. Wolfler-Calvo. Solving the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41 (4):470–483, 2007.

[49] O.M. Raft. A modular algorithm for an extended vehicle scheduling problem. *European Journal of Operational Research*, 11(1):67–76, 1982.

[50] G.K. Rand. Methodological choices in depot location studies. *Operational Research Quarterly*, 27(1):241–249, 1976.

[51] J. Renaud, F.F. Boctor, and G. Laporte. An improved petal heuristic for the vehicle routing problem. *Journal of the Operational Research Society*, 47(2):329–336, 1996.

[52] J. Renaud, G. Laporte, and F.F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers and Operations Research*, 23(3):229–235, 1996.

[53] P. Repoussis, D. Paraskevopoulos, C. Tarantilis, and G. Ioannou. A reactive greedy randomized variable neighborhood tabu search for the vehicle routing problem with time windows. *Hybrid Metaheuristics*, 4030: 124–138, 2006.

[54] S. Salhi and G.K. Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156, 1989.

[55] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.

[56] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Principles and Practice of Constraint Programming*, 1520:417–431, 1998.

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

[58] S.R. Thangiah and S. Salhi. Genetic clustering: An adaptive heuristic for the multidepot vehicle routing problem. *Applied Artificial Intelligence*, 15(4):361–83, 2001.

[59] C.J. Ting and C.H. Chen. A multiple ant colony optimization algorithm for the capacitated location routing problem. *International Journal of Production Economics*, 2012. doi: 10.1016/j.ijpe.2012.06.011. URL `http://dx.doi.org/10.1016/j.ijpe.2012.06.011`.

[60] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4): 333–346, 2003.

[61] D. Tuzun and L.I. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99, 1999.

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

[63] NA Wassan. A reactive tabu search for the vehicle routing problem. *Journal of the Operational Research Society*, 57(1):111–116, 2005.

[64] A. Wren and A. Holliday. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly*, 23(3):333–344, 1972.

[65] PC Yellow. A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly*, 21(2):281–283, 1970.

[66] V.F. Yu, S.W. Lin, W. Lee, and C.J. Ting. A simulated annealing heuristic for the capacitated location routing problem. *Computers and Industrial Engineering*, 58(2):288–299, 2010.