Chapter 8

VRP with Backhauls

Paolo Toth
Daniele Vigo

8.1 Introduction

In this chapter we consider the VRP with Backhauls (VRPB), also known as the linehaul-backhaul problem, an extension of the *Capacitated VRP* (CVRP) where the customer set is partitioned into two subsets. The first subset contains the *linehaul customers*, each requiring a given quantity of product to be delivered. The second subset contains the *backhaul customers*, where a given quantity of inbound product must be picked up.

This customer partition is extremely frequent in practical situations. A common example is that of the grocery industry, where supermarkets and shops are the linehaul customers and grocery suppliers are the backhaul customers. It has been widely recognized that in this mixed distribution–collection context a significant saving in transportation costs can be achieved by visiting backhaul customers in distribution routes (see, e.g., Golden et al. [21]).

More precisely, the VRPB can be stated as the problem of determining a set of vehicle routes visiting all customers, and (i) each vehicle performs one route; (ii) each route starts and finishes at the depot; (iii) for each route the total load associated with linehaul and backhaul customers does not exceed, separately, the vehicle capacity; (iv) on each route the backhaul customers, if any, are visited after all linehaul customers; and (v) the total distance traveled by the vehicles is minimized. The precedence constraint (iv) is practically motivated by the fact that vehicles are often rearloaded. Hence the on-board load rearrangement required by a mixed service is difficult, or impossible, to carry out at customer locations. Another practical reason is that, in many applications, linehaul customers have a higher service priority than backhaul customers.

We examine both the symmetric and asymmetric versions of VRPB. In the symmetric version of the problem (the VRPB) the distance between each pair of locations is the same in the two directions, whereas we have the asymmetric version (the AVRPB) when this assumption does not hold.

The problem can be formulated through the following graph theory model, where each customer corresponds to a vertex. Let $G = (V, A)$ be a complete undirected graph (directed for AVRPB) with vertex set $V := \{0\} \cup L \cup B$. Subsets $L = \{1, \ldots, n\}$ and $B = \{n+1, \ldots, n+m\}$ correspond to linehaul and backhaul customer subsets, respectively. A nonnegative quantity, d_j, of product to be delivered or collected (*demand*) is associated with each vertex $j \in V \setminus \{0\}$. Vertex 0 corresponds to the depot (with a fictitious demand $d_0 = 0$), where K identical vehicles with a given *capacity*, C, are stationed. Let c_{ij} be the nonnegative *cost* associated with arc $(i, j) \in A$, with $c_{ii} = +\infty$ for each $i \in V$ (and with $c_{ij} = c_{ji}$ for each $i, j \in V$ such that $i \neq j$ for VRPB). VRPB (and AVRPB as well) then consists of finding a min-cost collection of K simple circuits (vehicle routes) such that

(i) each circuit visits vertex 0;

(ii) each vertex $j \in V \setminus \{0\}$ is visited exactly once;

(iii) the sum of the demands of the linehaul and backhaul vertices visited by a circuit does not exceed, separately, the vehicle capacity C;

(iv) in each circuit the linehaul vertices precede the backhaul vertices, if any.

The objective is to minimize the total routing cost, defined as the sum of the costs of the arcs belonging to the circuits.

The mixed vehicle routes, which visit both linehaul and backhaul vertices, are implicitly oriented due to precedence constraint (iv). Let K_L (resp., K_B) denote the minimum number of vehicles needed to serve all the linehaul (resp., backhaul) vertices. To ensure feasibility, we assume that K (number of available vehicles) is not smaller than the minimum number of vehicles needed to serve all the vertices, i.e., $K \geq \max\{K_L, K_B\}$. The values K_L and K_B can be obtained by solving the Bin Packing Problem (BPP) instances associated with the linehaul and the backhaul vertices, respectively. In these instances the bin capacity is equal to C and each item j has a weight equal to the demand d_j of the corresponding vertex. Although BPP is NP-hard in the strong sense, many instances with hundreds of items can be optimally solved very effectively (see, e.g., Martello and Toth [26]).

As generally assumed in the literature, routes containing only backhaul vertices are not allowed. However, some of the heuristic approaches described in this chapter may be easily extended to consider the case where this limitation is not present.

VRPB and AVRPB are NP-hard in the strong sense, since they generalize the symmetric and asymmetric CVRP, respectively, arising when $B = \emptyset$.

Several heuristic algorithms for the solution of VRPB have been presented in the literature. Deif and Bodin [9] proposed an extension of the well-known Clarke and Wright [8] VRP heuristic, where the saving of the arcs connecting linehaul to backhaul customers is modified to delay the formation of mixed routes. Goetschalckx and Jacobs-Blecha [19] proposed an algorithm that builds initial routes with customers of the same type by using a space-filling curves heuristic; these routes are then merged to form the final set of vehicle

routes. It should be noted that the solutions obtained with both the Deif–Bodin and the space-filling heuristics could use more than K vehicles (hence leading to infeasible solutions). More recently, Goetschalckx and Jacobs-Blecha [20] described an extension to VRPB of the Fisher and Jaikumar [15] VRP heuristic. Toth and Vigo [34] proposed a different cluster-first, route-second heuristic for both VRPB and AVRPB. The algorithm starts from a (possibly infeasible) solution and tries to improve it through a local search procedure based on interroute and intraroute arc exchanges.

Yano et al. [36] proposed an exact, set-covering-based algorithm for the special case of VRPB in which the number of customers of each type in a circuit is not greater than four. Toth and Vigo [33] presented a branch-and-bound method for the exact solution of both VRPB and AVRPB. The algorithm is based on a reformulation of VRPB as an asymmetric problem, the addition of valid inequalities, and the computation of Lagrangian and additive lower bounds. Mingozzi, Giorgi, and Baldacci [28] proposed an exact approach for both VRPB and AVRPB. The method is based on a new set-partitioning model and the computation of an effective lower bound, obtained by considering different heuristic procedures for solving the dual of the LP relaxation of the original model.

Heuristic algorithms for the case in which the precedence constraint between linehaul and backhaul customers is not present are described by Golden et al. [21], Casco, Golden, and Wasil [5], Anily [2], and Salhi and Nagy [29]. The variant of VRPB where time window constraints are present was considered by Kontovradis and Bard [24], Duhamel, Potvin, and Rousseau [11], Gélinas et al. [17], and Thangiah, Potvin, and Sun [30] (see also section 7.6.2).

The chapter is organized as follows. In section 8.1.1, three classes of benchmark instances from the literature (two for VRPB and one for AVRPB) are illustrated. In section 8.2, two integer linear programming models for AVRPB (and hence for VRPB as well) are described: the two-index vehicle flow formulation proposed by Toth and Vigo [33] and the set-partitioning model proposed by Mingozzi, Giorgi, and Baldacci [28]. Section 8.3 presents the main relaxations used for deriving effective lower bounds on the optimal solution value of AVRPB. The exact enumerative algorithms for AVRPB proposed by Toth and Vigo [33] and by Mingozzi, Giorgi, and Baldacci [28], with the corresponding computational results on the set of benchmark instances, are described in section 8.4. The most effective heuristic algorithms for VRPB and AVRPB are summarized and computationally compared in section 8.5.

8.1.1 Benchmark Instances

Three classes of benchmark instances are used in the literature for experimentally comparing the performance of exact and heuristic algorithms proposed for VRPB and AVRPB. All instances described below are available on request from the authors.

The first class of test problem is made up by the 62 randomly generated Euclidean VRPB instances proposed by Goetschalckx and Jacobs-Blecha [19]. Customer coordinates are uniformly distributed in the interval [0, 24,000] for the x values and in the interval [0, 32,000] for the y values. The depot is located centrally in (12,000, 16,000). The cost c_{ij} of arc $(i, j) \in A$ is defined as the Euclidean distance between customers i and j. Customer demands are generated from a normal distribution with mean value 500 and standard deviation 200. Fourteen values for the total number of vertices, $n + m$ (from 25 to

150) are considered, with linehaul percentage equal to 50%, 66%, and 80%. For each value of $n + m$, the vehicle capacity is defined so that approximately 3 to 12 vehicles are used to serve all the demands.

The second class was proposed by Toth and Vigo [32] and contains 33 VRPB instances obtained from 11 CVRP test problems from the literature, with a total number of vertices between 21 and 100. For each CVRP test problem, three VRPB instances are generated, each corresponding to a linehaul percentage of 50%, 66%, and 80%. The customer set is partitioned by defining as backhaul the first customer in every two, three, or five, respectively, depending on the linehaul percentage desired. Customer demands and vehicle capacity are set equal to the original VRP values, and the number of available vehicles is defined by $K = \max\{K_L, K_B\}$. For the instances where $K_L < K_B$, linehaul and backhaul customer sets are swapped.

The third class contains 24 AVRPB instances (proposed by Toth and Vigo [33]) obtained from the real-world ACVRP instances described by Fischetti, Toth, and Vigo [13]. As in the second class, for each ACVRP instance three AVRPB instances are generated (corresponding to a linehaul percentage of 50%, 66%, and 80%, respectively), and the customer set is partitioned by defining as backhaul the first vertex in every two, three, and five, respectively. Customer demands, vehicle capacity, and cost matrix are equal to those of the original ACVRP. The values of K_L, K_B, and K are defined as for the second class.

8.2 Integer Linear Programming Models

The first formulation of the VRPB, proposed by Goetschalckx and Jacobs-Blecha [19], was an extension to VRPB of the nonlinear model of Fisher and Jaikumar [15]. We next present two integer linear programming models for AVRPB (hence, for VRPB as well), which are used to derive the relaxations on which the exact approaches are based.

8.2.1 Formulation of Toth and Vigo

The model proposed by Toth and Vigo [33] is based on the reformulation of VRPB as an asymmetric problem; thus it is valid for AVRPB as well. In the following we assume that single-customer (linehaul) routes are allowed.

Let us define $L_0 := L \cup \{0\}$ and $B_0 := B \cup \{0\}$. Let $\overline{G} = (\overline{V}, \overline{A})$ be a directed graph obtained from G by defining $\overline{V} = V$ and $\overline{A} = A_1 \cup A_2 \cup A_3$, where

$$A_1 := \{(i, j) \in A : i \in L_0, j \in L\},$$
$$A_2 := \{(i, j) \in A : i \in B, j \in B_0\},$$
$$A_3 := \{(i, j) \in A : i \in L, j \in B_0\}.$$

In other words, the arc set $\overline{A}$ can be partitioned into three disjoint subsets. The first subset, A_1, contains all the arcs from the depot and linehaul vertices to linehaul vertices. The second subset, A_2, contains all the arcs from backhaul vertices to backhaul vertices and the depot. The third subset, A_3, contains the so-called *interface arcs*, connecting linehaul vertices to backhaul vertices or the depot. Note that $\overline{A}$ does not contain arcs that cannot belong to a feasible solution. In fact, no arc from a backhaul to a linehaul vertex, or from the depot to a

backhaul vertex can belong to a feasible solution to VRPB, either because of the precedence constraint or because routes with only backhaul vertices are not allowed. Note that $\overline{A}$ is a proper subset of A and that each arc $(i, j) \in \overline{A}$ has a cost c_{ij} equal to the cost of the corresponding arc $(i, j) \in A$.

Let $\mathcal{L}$ (resp., $\mathcal{B}$) be the family of all subsets of vertices in L (resp., B), and let $\mathcal{F} = \mathcal{L} \cup \mathcal{B}$. For each $S \in \mathcal{F}$, let $r(S)$ be the minimum number of vehicles needed to serve all the customers in S. This value may be computed as the optimal solution value of the BPP with item set S and bin capacity equal to C. For each $i \in V$ let us define $\Delta_i^+ = \{j : (i, j) \in \overline{A}\}$ (i.e., the forward star of i) and $\Delta_i^- = \{j : (j, i) \in \overline{A}\}$ (i.e., the backward star of i). An integer linear programming formulation of VRPB and AVRPB is then

$$\text{(8.1)} \qquad \text{(P1)} \quad v(\text{P1}) = \min \sum_{(i,j)\in\overline{A}} c_{ij}\, x_{ij}$$

subject to

$$\text{(8.2)} \qquad \sum_{i\in\Delta_j^-} x_{ij} = 1 \qquad \forall\, j \in \overline{V} \setminus \{0\},$$

$$\text{(8.3)} \qquad \sum_{j\in\Delta_i^+} x_{ij} = 1 \qquad \forall\, i \in \overline{V} \setminus \{0\},$$

$$\text{(8.4)} \qquad \sum_{i\in\Delta_0^-} x_{i0} = K,$$

$$\text{(8.5)} \qquad \sum_{j\in\Delta_0^+} x_{0j} = K,$$

$$\text{(8.6)} \qquad \sum_{j\in S} \sum_{i\in\Delta_j^-\setminus S} x_{ij} \geq r(S) \qquad \forall\, S \in \mathcal{F},$$

$$\text{(8.7)} \qquad \sum_{i\in S} \sum_{j\in\Delta_i^+\setminus S} x_{ij} \geq r(S) \qquad \forall\, S \in \mathcal{F},$$

$$\text{(8.8)} \qquad x_{ij} \in \{0, 1\} \qquad \forall\, (i, j) \in \overline{A},$$

where $x_{ij} = 1$ if and only if arc (i, j) is in the optimal solution. Equations (8.2), (8.4) and (8.3), (8.5) impose indegree and outdegree constraints for the customer and the depot vertices, respectively. The so-called Capacity-Cut Constraints (CCCs) (8.6) and (8.7) impose both the connectivity and the capacity constraints. Note that because of degree constraints (8.2)–(8.5), for any given S the left-hand sides of (8.6) and (8.7) are equal (i.e., the number of arcs entering S is equal to the number of arcs leaving it). Hence, if constraints (8.6) are imposed, then constraints (8.7) are redundant and vice versa. Alternatively, an equivalent model for VRPB is obtained by imposing (8.6) only for each $S \in \mathcal{L}$ and (8.7) only for each $S \in \mathcal{B}$. Note also that in (8.6) and (8.7), the value $r(S)$ can be replaced by any lower bound on the optimal solution of the associated BPP (e.g., the continuous lower bound $\lceil d(S)/C \rceil$).

Finally, note that both families of constraints (8.6) and (8.7) have a cardinality growing exponentially with $\max\{n, m\}$, hence the LP relaxation of P1, defined by (8.1)–(8.7) and

$$0 \le x_{ij} \le 1, \qquad (i, j) \in \overline{A}, \tag{8.9}$$

cannot be directly solved, even for problems of moderate size.

For the symmetric version of the problem, the cost matrix associated with graph G is asymmetric due to the removal from G of the arcs connecting backhaul to linehaul vertices and the depot to backhaul vertices. However, the two submatrices corresponding to arcs with both endpoints in L_0 and in B, respectively, are symmetric.

8.2.2 Formulation of Mingozzi, Giorgi, and Baldacci

Mingozzi, Giorgi, and Baldacci [28] proposed the following set-partitioning model for AVRPB.

Let $G_L = (L_0, A_1)$ and $G_B = (B_0, A_2)$ denote the two subgraphs of $\overline{G}$ induced by the linehaul and the backhaul customers, respectively (see section 8.2.1). An elementary path P in G_L starting from the depot (resp., in G_B ending at the depot) is called feasible if its total demand satisfies the inequalities

$$C^L_{\min} \le \sum_{j \in P} d_j \le C \qquad \left(\text{resp., } C^B_{\min} \le \sum_{j \in P} d_j \le C \right), \tag{8.10}$$

where $C^L_{\min}$ (resp., $C^B_{\min}$) represents the minimum total demand of linehaul customers (resp., backhaul customers) of any feasible route. The values $C^L_{\min}$ and $C^B_{\min}$ can be computed as follows:

$$C^L_{\min} = \max \left\{ 0, \sum_{j \in L} d_j - (K-1)C \right\} \tag{8.11}$$

and

$$C^B_{\min} = \max \left\{ 0, \sum_{j \in B} d_j - (K-1)C \right\} \tag{8.12}$$

Let $t(P)$ denote both the terminal vertex of a feasible path P in G_L and the starting vertex of a feasible path P in G_B. Note that any pair of feasible paths P in G_L and P' in G_B (with P' possibly empty) and the arc $(t(P), t(P')) \in A_3$ (with $t(P') = 0$ when P' is empty) form a feasible route obtained by joining P and P' through the arc $(t(P), t(P'))$. Observe that the K routes of any feasible VRPB solution consist of K feasible paths in G_L, at least K_B feasible paths in G_B, and K arcs of A_3.

Let $\mathcal{L}$ be the index set of all feasible paths in G_L and let $\mathcal{L}_i \subseteq \mathcal{L}$ (resp., $\mathcal{L}^E_i \subseteq \mathcal{L}$) denote the index set of all paths passing through (resp., ending at) customer $i \in L$. Similarly, let $\mathcal{B}$ be the index set of all feasible paths in G_B and let $\mathcal{B}_i \subseteq \mathcal{B}$ (resp., $\mathcal{B}^S_i \subseteq \mathcal{B}$) denote the index set of all paths passing through (resp., starting from) customer $i \in B$. Finally, let $\overline{c}(\ell)$ denote the total cost of path P_ℓ (with $\ell \in \mathcal{L} \cup \mathcal{B}$), defined as the sum of the cost of its arcs.

The following binary variables are defined: x_ℓ, $\ell \in \mathcal{L}$, y_ℓ, $\ell \in \mathcal{B}$, and ξ_{ij}, $(i, j) \in A_3$, with $x_\ell = 1$, $y_{\ell'} = 1$, and $\xi_{ij} = 1$ if and only if the paths $\ell \in \mathcal{L}$, $\ell' \in \mathcal{B}$ and the arc $(i, j) \in A_3$ are in the optimal VRPB solution.

An integer programming formulation of the VRPB is

$$\text{(P2)} \quad v(\text{P2}) = \min \sum_{\ell \in \mathcal{L}} \overline{c}_\ell \, x_\ell + \sum_{\ell \in \mathcal{B}} \overline{c}_\ell \, y_\ell + \sum_{(i,j) \in A_3} c_{ij} \, \xi_{ij} \tag{8.13}$$

subject to

$$\sum_{\ell \in \mathcal{L}_i} x_\ell = 1 \qquad \forall \, i \in L, \tag{8.14}$$

$$\sum_{\ell \in \mathcal{B}_j} y_\ell = 1 \qquad \forall \, j \in B, \tag{8.15}$$

$$\sum_{\ell \in \mathcal{L}_i^E} x_\ell - \sum_{j \in B_0} \xi_{ij} = 0 \qquad \forall \, i \in L, \tag{8.16}$$

$$\sum_{\ell \in \mathcal{B}_j^S} y_\ell - \sum_{i \in L} \xi_{ij} = 0 \qquad \forall \, j \in B, \tag{8.17}$$

$$\sum_{(i,j) \in A_3} \xi_{ij} = K, \tag{8.18}$$

$$x_\ell \in \{0, 1\} \qquad \forall \, \ell \in \mathcal{L}, \tag{8.19}$$

$$y_\ell \in \{0, 1\} \qquad \forall \, \ell \in \mathcal{B}, \tag{8.20}$$

$$\xi_{ij} \in \{0, 1\} \qquad \forall \, (i, j) \in A_3. \tag{8.21}$$

Equations (8.14) and (8.15) require that each vertex $i \in L$ and $j \in B$ be visited by exactly one route. Equations (8.16) force the solution to contain an arc of A_3 starting from vertex $i \in L$ if a path in G_L ends at vertex i. Similarly, (8.17) requires an arc (i, j) with $i \in L$ and $j \in B$ if a path in G_B starts from vertex j. Equation (8.18) forces the solution to contain K routes by requiring K arcs of A_3.

Note that problem P2 (and its LP relaxation as well) cannot be solved directly, even for problems of moderate size, because the number of variables may be too large.

8.3 Relaxations

In this section, four relaxations for VRPB are presented. We first describe the three relaxations, based on model P1, proposed by Toth and Vigo [33]. The first relaxation is obtained by dropping the CCCs and leads to the solution of a *Transportation Problem* (TP). Then we describe a relaxation of the problem based on the projection of the feasible solution space, which requires the determination of degree-constrained *Shortest Spanning Trees* (SST) or *Arborescences* (SSA), as well as the optimal solution of min-cost flow problems. A Lagrangian lower bound is then derived by imbedding the relaxed degree constraints in the objective function. The lower bound is strengthened in a cutting-plane fashion by adding some of the previously relaxed CCCs. The Lagrangian lower bound is then combined, according to the additive approach, with the TP lower bound, thus obtaining an effective overall

bounding procedure. Finally, the bounding procedure proposed by Mingozzi, Giorgi, and Baldacci [28] is described. The lower bound is computed by combining different heuristic methods for solving the dual of the LP relaxation of model P2.

8.3.1 Relaxation Obtained by Dropping the CCCs

As described in Chapter 2, we may relax problem (8.1)–(8.8) by dropping CCCs (8.6) and (8.7). We thus obtain a TP, requiring the determination of a min-cost collection of circuits of G covering all the vertices in $V\backslash\{0\}$ once, and visiting the depot K times. This solution can be infeasible for VRPB since the total linehaul or backhaul customer demands of a circuit can exceed the vehicle capacity, or because of the possible existence of circuits not visiting the depot. Note that the precedence constraint between linehaul and backhaul vertices is satisfied because of the absence in $\overline{G}$ of arcs from backhaul to linehaul vertices or from the depot to backhaul vertices.

The solution of TP requires $O((n+m)^3)$ time through a transportation algorithm. However, in practice it is more effective to transform the problem into an *Assignment Problem* (AP) defined on the extended digraph obtained by adding $K-1$ copies of the depot to $\overline{G}$ (see section 1.3.2 for further details).

The computational experience showed that the quality of the lower bound provided by the TP relaxation is generally poor when symmetric instances are considered. However, this bound may be combined in an additive fashion, as shown in section 8.3.4, with the other bounds proposed in the following, leading to an effective overall bound.

8.3.2 Relaxation Based on Projection

The arcs of any feasible solution can be divided into three groups belonging to the previously defined subsets A_1, A_2, and A_3, respectively. As a consequence, problem P1 can be relaxed by dividing it into three independent subproblems, each relative to a different arc subset A_h, $h = 1, 2, 3$. A valid lower bound on v(P1) can be obtained by adding the three corresponding optimal solution values. This relaxation is used as the basis of the Lagrangian relaxation of VRPB described in the next section.

We now briefly describe the three resulting subproblems and discuss their efficient solution. The first subproblem, corresponding to the arc set A_1, is that of determining a min-cost collection of K capacitated disjoint simple paths, starting from the depot and spanning all the linehaul vertices. This problem is NP-hard in the strong sense and remains such even if we relax it by requiring the determination of a capacitated tree with fixed outdegree K at the depot and by spanning all the linehaul vertices. To obtain a polynomially solvable problem, the subproblem is further relaxed by dropping the capacity constraints. This leads to the determination of an SST with fixed degree K at the depot vertex (K-SST). Indeed, this problem (called P_1) can be efficiently solved, in $O(n^2)$ time, e.g., by using the algorithm proposed by Glover and Klingman [18]. If we consider the asymmetric version of VRPB, the only difference is that instead of determining trees, we must determine arborescences. Also in this case the capacitated SSA is an NP-hard problem (see, e.g., Toth and Vigo [31]). By dropping the capacity constraints we obtain the problem of determining an SSA with fixed outdegree K at the depot vertex (K-SSA), which can be solved in $O(n^2)$ time, e.g., by using the algorithm by Gabow and Tarjan [16].

Analogously, the second subproblem requires the determination of a min-cost collection of $\overline{K}$, with $K_B \leq \overline{K} \leq K_M = \min\{K, m\}$, disjoint capacitated paths entering the depot and spanning all the backhaul vertices. For any given $\overline{K}$, the subproblem can be relaxed in the same way as the previous one, thus requiring the determination of an SST with fixed degree $\overline{K}$ at the depot vertex. In the asymmetric case we must determine a *Shortest Spanning Antiarborescence* (SSAA) with fixed indegree $\overline{K}$ at the depot vertex ($\overline{K}$-SSAA). This problem (called $P_2(\overline{K})$) can be efficiently solved through the Gabow and Tarjan algorithm by transposing the corresponding cost matrix.

The third subproblem requires the determination of the so-called interface arcs, i.e., a min-cost collection of K arcs, $\overline{K}$ of which connect distinct linehaul vertices to distinct backhaul vertices (with $K_B \leq \overline{K} \leq K_M$), while the remaining $K - \overline{K}$ ones connect distinct linehaul vertices to the depot. For any given $\overline{K}$, this problem (called $P_3(\overline{K})$) can be efficiently solved by transforming it into an equivalent min-cost flow problem on an auxiliary layered network, which can be computed in $O(K(n+m)^2)$ time (see Ahuja, Magnanti, and Orlin [1]).

A valid lower bound on $v(P1)$ is then

$$LB = v(P_1) + \min_{K_B \leq \overline{K} \leq K_M} \left\{ v(P_2(\overline{K})) + v(P_3(\overline{K})) \right\}. \tag{8.22}$$

By using parametric techniques, the computation of $v(P_2(\overline{K}))$ and $v(P_3(\overline{K}))$ for all values of $\overline{K}$ can be performed in $O(m^2)$ and in $O((2K - K_B)(n+m)^2)$ time, respectively. Hence, the overall time complexity of the computation of LB is $O((2K - K_B)(n+m)^2)$.

8.3.3 Lagrangian Relaxation

Lower bound LB of the previous section can be strengthened by considering some of the removed constraints and by introducing them in a Lagrangian fashion into the objective function. In particular, the indegree constraints (8.2) are considered only for the backhaul vertices ($j \in B$), and the outdegree constraints (8.3) only for the linehaul vertices ($i \in L$). Let λ be the vector of the corresponding $n + m$ Lagrangian multipliers.

In addition, let $\mathcal{F}_1 \subset \mathcal{L}$ and $\mathcal{F}_2 \subset \mathcal{B}$ be two given (small cardinality) families of the exponentially many subsets contained in $\mathcal{F}$. CCCs (8.6) and (8.7) are imposed only for the subsets of $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively,

Families $\mathcal{F}_1$ and $\mathcal{F}_2$ are determined, in a branch-and-cut fashion, by considering vertex subsets for which the associated CCC is violated by the previous relaxations. The violated inequalities are introduced into the objective function with nonnegative Lagrangian multiplier vectors π and ρ, respectively. Good Lagrangian multipliers λ, π, and ρ are computed through a two-level subgradient optimization procedure. A similar approach was applied by Fisher [14] to the CVRP, by Malik and Yu [25] to the capacitated SST, and by Toth and Vigo [31] to the capacitated SSA.

Note that given arbitrary multipliers, the corresponding Lagrangian cost matrix could be completely asymmetric even if the original matrix contained symmetric submatrices, as happens for VRPB. Hence, only the asymmetric version of the problem is considered in the following.

A polynomial time exact procedure is used to determine infeasible vertex subsets, if any, such that the current Lagrangian problem solution, $\overline{x}$, violates the associated CCCs (8.6) or (8.7).

To separate violated CCCs of type (8.6) for $S \in \mathcal{L}$, the arborescence corresponding to arc subset $W_L = \{(i, j) \in A_1 : \overline{x}_{ij} = 1\}$ is considered. For each linehaul vertex i, let X_i be the subset containing all the linehaul vertices belonging to the subarborescence in W_L rooted at i (with $i \in X_i$), and let w_i denote the total demand of subset X_i. If $w_i > C$, then constraint (8.6) for $S = X_i$ is violated, since $r(X_i)$ is greater than 1 and only one arc in W_L enters subset X_i. Hence, subset X_i is added to family $\mathcal{F}_1$. The computation of w_i for all $i \in L$ can be performed in $O(n)$ time. Note that although only vertex sets associated with subarborescences whose arcs are in W_L are considered, it is easy to see that if a violated constraint of type (8.6) exists, then it is detected by the above procedure.

Analogously, violated CCCs of type (8.7) can be separated by considering only the antiarborescence corresponding to arcs subset $W_B = \{(i, j) \in A_2 : \overline{x}_{ij} = 1\}$, thus determining vertex subsets to be added to family $\mathcal{F}_2$.

8.3.4 Overall Additive Lower Bound

The lower bounds obtained through the Lagrangian and TP relaxations can be combined according to the *additive approach* proposed by Fischetti and Toth [12]. This approach allows for the combination of different lower bounding procedures, each exploiting different substructures of the problem. When applied to VRPB, each procedure returns a lower bound δ and a *residual cost matrix*, $\tilde{c}$, such that

$$\begin{aligned} &\tilde{c} \geq 0 \\ &\delta + \tilde{c}x \leq cx \quad \forall\, x \text{ satisfying (8.1)–(8.8).} \end{aligned}$$

The entries of $\tilde{c}$ represent lower bounds on the increment of the optimal solution value if the corresponding arc is imposed in the solution. The bounding procedures are applied in sequence and each of them uses the *residual cost matrix* returned by the previous procedure (the first procedure starts with the original cost matrix). The additive lower bound is the sum of the lower bounds obtained by each procedure. For further details, see also Fischetti, Toth, and Vigo [13].

It can be shown that the Lagrangian modified costs, as well as the reduced costs of the TP relaxation of section 8.3.1, are valid residual costs. At the end of the computation of the Lagrangian lower bound, the modified cost matrix $\overline{c}$ is extended by adding $K - 1$ rows and columns corresponding to the copies of the depot, and the resulting TP relaxation is solved, returning the lower bound increment. The final TP reduced costs can be used for reduction purposes.

8.3.5 Relaxation Based on the Set-Partitioning Model

Mingozzi, Giorgi, and Baldacci [28] described a heuristic procedure that finds a feasible solution of the dual problem D2 of the linear relaxation of P2 (see section 8.2.2), thus pro-

viding a valid lower bound to VRPB. This procedure does not require the explicit generation of the path sets $\mathcal{L}$ and $\mathcal{B}$. Moreover, this dual solution is used to reduce drastically the sets $\mathcal{L}$ and $\mathcal{B}$ by removing, through effective reduction procedures, those paths that cannot belong to any optimal VRPB solution.

Let u_i, $i \in L$, and v_j, $j \in B$, be the dual variables associated with constraints (8.14) and (8.15), respectively. Moreover, denote with α_i, $i \in L$, and β_j, $j \in B$, the dual variables associated with constraints (8.16) and (8.17), respectively. Finally, let w be the dual variable associated with constraint (8.18). The dual of the LP relaxation of P2 is

$$\text{(D2)} \quad v(\text{D2}) = \max \sum_{i \in L} u_i + \sum_{j \in B} v_j + K w \tag{8.23}$$

subject to

$$\sum_{k \in P_\ell} u_k + \alpha_i \le \bar{c}_\ell \qquad \forall\, \ell \in \mathcal{L}_i^E,\ i \in L, \tag{8.24}$$

$$\sum_{k \in P_\ell} u_k + \beta_j \le \bar{c}_\ell \qquad \forall\, \ell \in \mathcal{B}_j^S,\ j \in B, \tag{8.25}$$

$$-\alpha_i - \beta_j + w \le c_{ij} \qquad \forall\, (i, j) \in A_3, \tag{8.26}$$

$$u_i, \alpha_i \text{ unrestricted} \qquad \forall\, i \in L, \tag{8.27}$$

$$v_j, \beta_j \text{ unrestricted} \qquad \forall\, j \in B, \tag{8.28}$$

$$w \text{ unrestricted} \tag{8.29}$$

with $\beta_0 = 0$ in the dual constraints (8.26) for each $(i, 0) \in A_3$.

In Mingozzi, Giorgi, and Baldacci [28], a heuristic procedure, called HDS, is proposed for computing a feasible solution to D2. The procedure is based on the additive bounding method introduced by Fischetti and Toth [12] for combinatorial optimization problems. Similar procedures were used by Bianco, Mingozzi, and Ricciardelli [4], Hadjiconstantinou, Christofides, and Mingozzi [22], and Mingozzi et al. [27] for solving the multiple-depot vehicle-scheduling problem, the CVRP, and the crew-scheduling problem, respectively. This procedure computes a feasible solution to D2 as the sum of the dual solutions obtained by a sequence of different relaxations of P2, where each relaxation is based on a different substructure of the problem. Procedure HDS is based on the following general idea. A feasible solution $\pi = \pi^1 + \pi^2 + \cdots + \pi^t$ of the linear problem

$$\text{(LP)} \quad \max\ \pi b \tag{8.30}$$

subject to

$$\pi A \le c, \tag{8.31}$$

$$\pi \text{ unrestricted} \tag{8.32}$$

can be obtained by solving a sequence of t linear programs $\text{LP}^1, \text{LP}^2, \ldots, \text{LP}^t$ by using t different heuristic procedures $\text{H}^1, \text{H}^2, \ldots, \text{H}^t$. Procedure H^r ($r = 1, 2, \ldots, t$), finds a

feasible solution π^r of the linear program LP^r defined as

$$(\mathrm{LP}^r) \quad \max\ \pi^r b \tag{8.33}$$

subject to

$$\pi^r A \le c, \tag{8.34}$$

$$\pi^r \text{ unrestricted}, \tag{8.35}$$

where $c^r = c - (\pi^0 + \pi^1 + \cdots + \pi^{r-1})A$ and $\pi^0 = 0$ (with $\mathrm{LP}^1 = \mathrm{LP}$).

Procedure HDS involves two heuristic procedures, H^1 and H^2, used in sequence. Procedure H^1 finds a feasible solution $(u^1, v^1, \alpha^1, \beta^1, w^1)$ of problem D^1, which coincides with D2, without requiring the generation of the sets $\mathcal{L}$ and $\mathcal{B}$. The second procedure, H^2, solves problem D^2, which is obtained from D2 by replacing the path costs $\overline{c}_\ell$, $\ell \in \mathcal{L} \cup \mathcal{B}$, and the arc costs c_{ij}, $(i, j) \in A_3$, with the reduced costs $\overline{c}'_\ell$ and c'_{ij} computed according to the solution $(u^1, v^1, \alpha^1, \beta^1, w^1)$ obtained by procedure H^1. Procedure H^2 requires the generation of limited subsets of the sets $\mathcal{L}$ and $\mathcal{B}$.

Procedure H^1 is based on the observation that any route of a feasible solution of VRPB contains an arc of set A_3. A lower bound for VRPB can be obtained as follows. Associate with each arc $(i, j) \in A_3$ a modified cost representing a lower bound on the cost of the least-cost route passing through it. As a consequence, the sum of the modified costs of the K vertex-disjoint arcs of minimum cost of A_3 is a valid lower bound for VRPB. This problem is called $PR(\lambda, \mu)$. Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ and $\mu = (\mu_1, \mu_2, \ldots, \mu_m)$ be two vectors associated with the linehaul and backhaul customers, respectively. Let each arc $(i, j) \in \overline{A}$ be associated with a cost $\tilde{c}_{ij}$, defined as

$$\tilde{c}_{ij} = \begin{cases} c_{ij} - \lambda_j & \text{if } j \in L, \\ c_{ij} - \mu_j & \text{if } j \in B, \\ c_{ij} & \text{if } j = 0. \end{cases} \tag{8.36}$$

The modified costs associated with the arcs in A_3 are determined through the so-called least-cost q-paths (see Christofides, Mingozzi, and Toth [7]), computed with respect to costs $(\tilde{c}_{ij})$. Moreover, for given vectors λ and μ, the dual of the LP relaxation of $\mathrm{PR}(\lambda, \mu)$ can be efficiently computed in $O((n + m)^3)$ time by transforming $\mathrm{PR}(\lambda, \mu)$ into a transportation problem. Good vectors λ and μ are obtained by using a subgradient optimization procedure.

As for procedure H^2, let $(u^1, v^1, \alpha^1, \beta^1, w^1)$ be the feasible solution of D^1 of value $v(\mathrm{D}^1)$, produced by procedure H^1. The reduced costs of the variables of problem P2 are given by

$$\overline{c}'_\ell = \overline{c}_\ell - \sum_{k \in P_\ell} u^1_k - \alpha^1_{t_\ell} \qquad \forall\, \ell \in \mathcal{L}, \tag{8.37}$$

$$\overline{c}'_\ell = \overline{c}_\ell - \sum_{k \in P_\ell} v^1_k - \beta^1_{t_\ell} \qquad \forall\, \ell \in \mathcal{B}, \tag{8.38}$$

$$c'_{ij} = c_{ij} - \alpha^1_i - \beta^1_j - w^1 \qquad \forall\, (i, j) \in A_3. \tag{8.39}$$

Let D^2 denote the problem obtained from D2 by replacing $\{\overline{c}_\ell\}$ with $\{\overline{c}'_\ell\}$ and $\{c_{ij}\}$ with $\{c'_{ij}\}$. Problem D^2 cannot be solved directly because the number of constraints may be too large.

Mingozzi, Giorgi, and Baldacci [28] described a heuristic procedure, called H^2, for reducing the number of constraints of D^2 so that the resulting problem, called $\overline{D}^2$, can be solved directly, and any solution of $\overline{D}^2$ is a feasible solution of D^2. Problem $\overline{D}^2$ is obtained from D^2 as follows:

(i) Reduce the number of constraints (8.24) and (8.25) by replacing $\mathcal{L}$ and $\mathcal{B}$ with subsets $\overline{\mathcal{L}} \subseteq \mathcal{L}$ and $\overline{\mathcal{B}} \subseteq \mathcal{B}$, respectively, of limited size.

(ii) Add constraints to force any solution of $\overline{D}^2$ to satisfy constraints (8.24) for any $\ell \in \mathcal{L} \setminus \overline{\mathcal{L}}$ and constraints (8.25) for any $\ell \in \mathcal{B} \setminus \overline{\mathcal{B}}$.

Let $\overline{\mathcal{L}} \subseteq \mathcal{L}$ and $\overline{\mathcal{B}} \subseteq \mathcal{B}$, be the subsets of paths satisfying the conditions

- $\max_{\ell \in \overline{\mathcal{L}}}[\overline{c}'_\ell] \leq \min_{\ell \in \mathcal{L} \setminus \overline{\mathcal{L}}}[\overline{c}'_\ell]$,
- $\max_{\ell \in \overline{\mathcal{B}}}[\overline{c}'_\ell] \leq \min_{\ell \in \mathcal{B} \setminus \overline{\mathcal{B}}}[\overline{c}'_\ell]$,
- $\overline{c}'_\ell < UB - v(D^1), \quad \ell \in \overline{\mathcal{L}} \cup \overline{\mathcal{B}}$,

and

$$|\overline{\mathcal{L}}| \leq \text{ Maxsize}, \tag{8.40}$$

$$|\overline{\mathcal{B}}| \leq \text{ Maxsize}, \tag{8.41}$$

where UB is the cost of a feasible solution of VRPB and Maxsize is a given positive integer. In addition, set $\overline{\mathcal{L}}_i^E = \overline{\mathcal{L}} \cap \mathcal{L}_i^E$, $i \in L$, and $\overline{\mathcal{B}}_j^S = \overline{\mathcal{B}} \cap \mathcal{B}_j^S$, $j \in B$. Subsets $\overline{\mathcal{L}}$ and $\overline{\mathcal{B}}$ are generated through a dynamic programming procedure called GEN$\mathcal{P}$.

The reduced problem $\overline{D}^2$ is obtained form D^2 by replacing $\mathcal{L}_i^E$ and $\mathcal{B}_j^S$ with $\overline{\mathcal{L}}_i^E$ and $\overline{\mathcal{B}}_j^S$, respectively, and by adding the following constraints:

$$u_i + \delta_i \leq U_i \qquad \forall\, i \in L, \tag{8.42}$$

$$\alpha_i - \delta_i \leq 0 \qquad \forall\, i \in L, \tag{8.43}$$

$$v_j + \theta_j \leq V_j \qquad \forall\, j \in B, \tag{8.44}$$

$$\beta_j - \theta_j \leq 0 \qquad \forall\, j \in B, \tag{8.45}$$

$$\delta_i \geq 0 \qquad \forall\, i \in L, \tag{8.46}$$

$$\theta_j \geq 0 \qquad \forall\, j \in B. \tag{8.47}$$

As shown by Mingozzi, Giorgi, and Baldacci [28], constraints (8.42)–(8.45) ensure that any solution of $(\overline{D}^2)$ is a feasible solution of (D^2), with the same objective function value, if the upper bounds U_i and V_j are computed as

$$U_i = d_i \hat{c}^L / D, \quad i \in L,$$

and

$$V_j = d_j \hat{c}^B / D, \quad j \in B,$$

where $\hat{c}^L = \max_{\ell \in \overline{\mathcal{L}}}[\overline{c}'_\ell]$ and $\hat{c}^B = \max_{\ell \in \overline{\mathcal{B}}}[\overline{c}'_\ell]$.

Procedure H^2 determines the optimal solution $(u^2, v^2, \alpha^2, \beta^2, w^2)$ of $(\overline{D}^2)$ (and hence of D^2 as well) by optimally solving, through an LP solver, the corresponding dual problem.

Concluding, procedure HDS finds a solution $(u', v', \alpha', \beta', w')$ of D2 of cost $v'(D2) = v(D^1) + v(\overline{D}^2)$ by setting $u' = u^1 + u^2$, $v' = v^1 + v^2$, $\alpha' = \alpha^1 + \alpha^2$, $\beta' = \beta^1 + \beta^2$, $w' = w^1 + w^2$.

8.4 Exact Algorithms

We next describe the exact algorithms, proposed by Toth and Vigo [33] and by Mingozzi, Giorgi, and Baldacci [28], for finding the optimal solution to AVRPB (sections 8.4.1 and 8.4.2, respectively). The performance of these algorithms is analyzed through computational experiments in section 8.4.3.

8.4.1 Algorithm of Toth and Vigo

The lowest-first branch-and-bound algorithm proposed by Toth and Vigo [33] is based on the Lagrangian relaxation described in section (8.3.3). The branching rule is an extension of the well-known *subtour elimination* scheme.

Let ν be the current node of the branch decision tree and let I_ν and F_ν denote the set of arcs imposed and excluded in the current solution, respectively. By construction, I_ν induces a collection of feasible routes and feasible paths, some of which may correspond to isolated vertices. A restricted AVRPB instance associated with I_ν and F_ν is constructed by means of the following preprocessing phase. First, all the vertices covered in I_ν by complete routes are removed from the graph, and the number $\widetilde{K}$ of available vehicles is updated accordingly. Then, each path induced by the arcs of I_ν not incident in vertex 0, if any, is shrunken into a single vertex. The demand associated with the new vertex is the sum of the demands of the vertices belonging to the shrunken path. A new restricted cost matrix and new demands for the shrunken vertices are thus obtained. The arcs $(0, j) \in I_\nu$ (resp., $(j, 0) \in I_\nu$) are imposed by preventing all other arcs from entering (resp., leaving) vertex j. Finally all the arcs in F_ν are excluded by setting the cost of the corresponding arc in the restricted matrix to $+\infty$.

After the preprocessing phase, the overall bounding procedure (see section 8.3.4) is applied, and the best solution $\overline{x}$ of the Lagrangian problem (reconstructed for the original graph) is stored. If $\overline{x}$ is feasible for AVRPB, the current incumbent solution is possibly updated. Otherwise, a sequence $Q := \{(v_1, v_2), (v_2, v_3), \ldots, (v_h, v_{h+1})\}$ to branch with is determined. The sequence is chosen as the subset of $\{(i, j) \in \overline{A} : \overline{x}_{ij} = 1\}$ having the minimum number of nonimposed arcs, which defines either a path which is infeasible (i.e., it is either a cycle disconnected from the depot or a path in which the total demand of linehaul and backhaul vertices exceed, separately, the vehicle capacity) or, if $\overline{x}$ is feasible, a circuit through vertex 0 (in this case, $v_1 = v_{h+1} = 0$). h descendant nodes $\nu_i, i = 1, \ldots, h$, are then generated:

$$I_{\nu_i} := I_\nu \cup \{(v_1, v_2), \ldots, (v_{i-1}, v_i)\},$$
$$F_{\nu_i} := F_\nu \cup \{(v_i, v_{i+1})\}$$

(with $I_{v_1} := I_v$). When Q is a feasible circuit, an additional son node, v_{h+1}, with $I_{v_{h+1}} := I_v \cup Q$ and $F_{v_{h+1}} := F_v$, is generated. Clearly, nodes v_i such that $I_{v_i} \cap F_{v_i} \neq \emptyset$ are not generated.

As for the computation of the lower bounds, at the root node of the branch-decision tree a large number of iterations of the subgradient optimization procedure are performed to obtain the best possible lower bound value. At the other nodes of the branch-decision tree a much smaller number of iterations are performed. At each node of the branch-decision tree, the families $\mathcal{F}_1$ and $\mathcal{F}_2$, the associated multipliers μ and ρ, and the multipliers λ are initialized with the sets and multiplier values corresponding to the best Lagrangian lower bound obtained at the father node.

The performance of the algorithm is enhanced by means of reduction and dominance procedures and through feasibility checks. Moreover, at each node of the branch-decision tree, if the node is not fathomed by the lower bound or by the dominance rules, the heuristic algorithm HTV (see section 8.5.3) is applied, for possible updating of the incumbent solution.

8.4.2 Algorithm of Mingozzi, Giorgi, and Baldacci

The exact method proposed by Mingozzi, Giorgi, and Baldacci [28], called EHP, consists of reducing the number of variables of the integer program P2 so that the resulting problem can be solved by an integer programming solver. This technique is in some way similar to the so-called *core problem* approach, used to solve large-size NP-hard problems.

Let $(u', v', \alpha', \beta', w')$ be the solution of D2 of cost $v'(\mathrm{D2})$ obtained by procedure HDS, and let $\overline{c}'_\ell$, $\ell \in \mathcal{L} \cup \mathcal{B}$, and c'_{ij}, $(i, j) \in A_3$, be the reduced costs corresponding to this dual solution. It is well known that an optimal solution of VRPB can be obtained by replacing in problem P2 the sets $\mathcal{L}$, $\mathcal{B}$, and A_3 with the subsets $\mathcal{L}' \subseteq \mathcal{L}$, $\mathcal{B}' \subseteq \mathcal{B}$, and $A'_3 \subseteq A_3$, defined as

$$\mathcal{L}' = \{\ell : \ell \in \mathcal{L}, \overline{c}'_\ell < UB - v'(\mathrm{D2})\},$$
$$\mathcal{B}' = \{\ell : \ell \in \mathcal{B}, \overline{c}'_\ell < UB - v'(\mathrm{D2})\},$$
$$A'_3 = \{(i, j) : (i, j) \in A_3, c'_{ij} < UB - v'(\mathrm{D2})\}.$$

However, the size of $\mathcal{L}'$ and/or $\mathcal{B}'$ may be too large to allow for the direct solution of the corresponding problem. Hence, Mingozzi, Giorgi, and Baldacci [28] proposed to generate $\mathcal{L}'$ and $\mathcal{B}'$ so that their size is limited and the resulting problem P2$'$ is solvable. Subsets $\mathcal{L}'$ and $\mathcal{B}'$ are generated by means of the same algorithm GEN$\mathcal{P}$ used by procedure H^2 (see section 8.3.5), where $v(\mathrm{D}^1)$ is replaced by $v'(\mathrm{D2})$. Note that the size of each set $\mathcal{L}'$ and $\mathcal{B}'$ is limited by the value Maxsize used in algorithm GEN$\mathcal{P}$. Let (x^*, y^*, ξ^*) be an optimal solution of P2$'$ of cost $v(\mathrm{P2}')$ (we assume $v(\mathrm{P2}') = +\infty$ if subsets $\mathcal{L}'$ and $\mathcal{B}'$ contain no feasible solution of VRPB). If $v(\mathrm{P2}') < +\infty$, then solution (x^*, y^*, ξ^*) is a feasible solution of VRPB, and it also may be an optimal one. Let

$$\Delta = \min\left\{\max_{\ell \in \mathcal{L}'}[\overline{c}'_\ell], \max_{\ell \in \mathcal{B}'}[\overline{c}'_\ell]\right\}.$$

We have the following cases:

(i) $v(\mathrm{P2}') \leq v'(\mathrm{D2}) + \Delta$. In this case the optimal solution of P2′ is also an optimal solution of VRPB. This derives from the property that any solution of VRPB involving at least one path of $\mathcal{L} \setminus \mathcal{L}'$ or $\mathcal{B} \setminus \mathcal{B}'$ has a cost greater than or equal to $v'(\mathrm{D2}) + \Delta$.

(ii) $v(\mathrm{P2}') > v'(\mathrm{D2}) + \Delta$. The optimal solution of P2′ might not be an optimal solution of VRPB. However, it is easy to note that, in this case, the value $v'(\mathrm{D2}) + \Delta$ is a valid lower bound to any optimal solution of VRPB.

The optimal solution of P2′ is obtained by means of the integer programming code CPLEX 3.0. Note that the method may terminate, under certain circumstances (see case (ii) above), without having found an optimal solution.

Nothing is explicitly said by Mingozzi, Giorgi, and Baldacci [28] about the steps to be performed to overcome this drawback. Following what is proposed for other core problem techniques, a possible approach could be to iteratively increase the value of parameter Maxsize and to repeat the procedure until the optimality of the current solution of P2′ can be proved, or subsets $\mathcal{L}'$ and $\mathcal{B}'$ coincide with $\mathcal{L}$ and $\mathcal{B}$, respectively.

8.4.3 Computational Results for the Exact Algorithms

The branch-and-bound algorithm of Toth and Vigo (see section 8.4.1), called TV herein, was implemented in FORTRAN and run on a Pentium 60 MHz personal computer (5.3 Mflops according to Dongarra [10]) on the three classes of problem instances described in section 8.1.1.

The exact algorithm EHP of Mingozzi, Giorgi, and Baldacci (see section 8.4.2) has been coded in FORTRAN 77 and run on a Silicon Graphics Indy MIPS R4400/200 MHz processor (30 Mflops; see Dongarra [10]) on the first two classes of test problems. According to Mingozzi, Giorgi, and Baldacci, the Pentium 60 MHz used for TV is about four times slower than the SGI used for EHP. Package CPLEX 3.0 has been used as the LP solver in procedure H^2 and as the integer programming solver in EHP. The parameter Maxsize, used in GEN$\mathcal{P}$, has been set to 70,000 in both procedures H^2 and EHP.

Tables 8.1, 8.2, and 8.3 report the results obtained by algorithms TV and EHP on the instances of the corresponding classes. For the first class of VRPB instances, the cost c_{ij} of arc $(i, j) \in A$ is defined as the Euclidean distance between customers i and j, multiplied by 10 and rounded to the nearest integer. The values reported in Table 8.1 are those obtained using the integer matrix c, divided by 10 and rounded to the nearest integer. Algorithm TV has been run on the first 34 VRPB instances of the first class (with a time limit of 6000 seconds for each instance), on the first 30 VRPB instances of the second class (with a time limit of 18,000 seconds), and on all the 24 AVRPB instances of the third class (with a time limit of 6000 seconds). Algorithm EHP has been run on the first 47 VRPB instances of the first class and on all the 33 VRPB instances of the second class (with a time limit of 25,000 seconds for each instance of both classes). No result is given by Mingozzi, Giorgi, and Baldacci [28] on the AVRPB instances of the third class.

For each problem we give the problem name, the problem size (namely, the values of n and m), the available number of vehicles K, and the minimum number of vehicles needed to serve the linehaul and the backhaul vertices, K_L and K_B, respectively. The values of K_L

Table 8.1. *Behavior of the exact algorithms on the VRPB instances of the first class. Computing times in Pentium* 60 *MHz seconds (time limit of* 6000 *seconds) for TV and in SGI* R4400/200 *MHz seconds (time limit of* 25,000 *seconds) for EHP.*

						TV				EHP			
Name	n	m	K	K_L	K_B	% LB		z^*	Time	% LB		z^*	Time
A1	20	5	8	7	2	98.3		229886	902	98.8		229886	5
A2	20	5	5	4	1	98.1		180119	209	98.8		180119	4
A3	20	5	4	3	1	100.0		163405	3	100.0		163405	10
A4	20	5	3	3	1	100.0		155796	3	100.0		155796	12
B1	20	10	7	7	4	96.0		239080	148	97.8		239080	14
B2	20	10	5	4	3	97.4		198048	151	97.9		198048	40
B3	20	10	3	3	2	100.0		169372	1	100.0		169372	4
C1	20	20	7	6	6	95.7		249448	227	98.2		249448	17
C2	20	20	5	4	4	96.5		215020	322	97.0		215020	18
C3	20	20	5	3	3	99.8		199346	84	100.0		199346	25
C4	20	20	4	3	3	100.0		195366	5	100.0		195366	25
D1	30	8	12	10	3	97.0		322530	289	98.8		322530	9
D2	30	8	11	10	3	94.5		316709	491	98.2		316709	13
D3	30	8	7	6	2	95.9	*	239479	—	96.8		239479	51
D4	30	8	5	5	2	95.4	*	205832	—	96.3		205832	161
E1	30	15	7	6	3	95.1		238880	476	100.0		238880	12
E2	30	15	4	4	2	97.9		212263	788	100.0		212263	41
E3	30	15	4	3	2	98.2		206659	482	98.9		206659	64
F1	30	30	6	5	6	96.6		263173	756	97.4		263173	2049
F2	30	30	7	5	6	98.3		265213	891	98.9		265213	44
F3	30	30	5	4	4	98.0		241120	468	98.8		241120	76
F4	30	30	4	3	3	97.3		233861	3523	97.3		233861	173
G1	45	12	10	9	3	91.3	*	307274	—	97.8		306305	3556
G2	45	12	6	6	2	93.3	*	245441	—	98.8		245441	229
G3	45	12	5	5	2	96.2		229507	4225	97.3	*	229507	967
G4	45	12	6	5	2	96.7	*	233184	—	97.5	*	232521	89
G5	45	12	5	4	1	97.9		221730	3433	98.0	*	221730	157
G6	45	12	4	3	1	96.6		213457	840	97.0	*	213457	103
H1	45	23	6	6	3	96.6		268933	1344	98.4	*	268933	454
H2	45	23	5	5	3	99.4		253365	5020	99.5		253365	221
H3	45	23	4	4	2	99.2		247449	1465	99.4		247449	177
H4	45	23	5	4	2	99.7		250221	1287	99.6		250221	179
H5	45	23	4	3	2	99.3		246121	406	99.3		246121	277
H6	45	23	5	3	2	99.4		249135	416	99.5		249135	173
I1	45	45	10	8	9			n.a.		97.0	*	353021	20225
I2	45	45	7	6	7			n.a.		98.7		309943	6395
I3	45	45	5	4	5			n.a.		96.7	*	294833	18045
I4	45	45	6	4	5			n.a.		97.7	*	295988	20055
I5	45	45	7	4	5			n.a.		98.2	*	301226	8642
J1	75	19	10	10	3			n.a.		98.3	*	335006	1640
J2	75	19	8	8	2			n.a.		94.7	*	315644	218
J3	75	19	6	5	2			n.a.		96.2	*	282447	363
J4	75	19	7	7	2			n.a.		94.9	*	300548	260
K1	75	38	10	10	5			n.a.		97.6	*	394637	—
K2	75	38	8	8	4			n.a.		98.6	*	362360	2618
K3	75	38	9	8	4			n.a.		98.5	*	365693	3812
K4	75	38	7	7	3			n.a.		95.2	*	358308	265

*Optimality not proved.

Table 8.2. *Behavior of the exact algorithms on the VRPB instances of the second class. Computing times in Pentium* 60 *MHz seconds (time limit of* 18,000 *seconds) for TV and in SGI* R4400/200 *MHz seconds (time limit of* 25,000 *seconds) for EHP.*

					TV				EHP			
Name	n	m	K	K_B	% LB		z^*	Time	% LB		z^*	Time
eil22_50	11	10	3	2	100.0		371	3	100.0		371	6
eil22_66	14	7	3	1	100.0		366	6	100.0		366	3
eil22_80	17	4	3	1	98.9		375	55	99.2		375	6
eil23_50	11	11	2	1	100.0		682	2	100.0		682	1
eil23_66	15	7	2	1	98.8		649	65	99.4		649	7
eil23_80	18	4	2	2	98.1		623	36	98.7		623	9
eil30_50	15	14	2	2	100.0		501	3	100.0		501	8
eil30_66	20	9	3	1	98.5		537	119	97.6		537	17
eil30_80	24	5	3	1	100.0		514	13	97.9		514	31
eil33_50	16	16	3	2	98.4		738	292	100.0		738	46
eil33_66	22	10	3	1	94.8		750	1338	100.0		750	27
eil33_80	26	6	3	1	93.9		736	1655	99.3		736	44
eil51_50	25	25	3	3	99.3		559	441	99.6		559	66
eil51_66	34	16	4	2	97.8		548	2754	99.3		548	68
eil51_80	40	10	4	1	98.0		565	4436	98.1		565	691
eilA76_50	37	38	6	5	98.2		739	15931	99.2		739	884
eilA76_66	50	25	7	4	95.4		768	13464	99.0		768	1205
76_80	60	15	8	2	90.5	*	781	—	97.7	*	781	596
eilB76_50	37	38	8	7	97.6		801	16345	99.3		801	124
eilB76_66	50	25	10	5	91.2		873	12990	99.0		873	2918
eilB76_80	60	15	12	3	85.2		919	10414	99.5		919	821
eilC76_50	37	38	5	4	98.9		713	10344	98.9		713	16659
eilC76_66	50	25	6	3	97.6	*	734	—	99.2		734	952
eilC76_80	60	15	7	2	93.7	*	733	—	97.8	*	733	—
eilD76_50	37	38	4	3	99.7		690	401	99.7		690	197
eilD76_66	50	25	5	2	98.5	*	715	—	98.6	*	715	5023
eilD76_80	60	15	6	2	96.8	*	703	—	99.0	*	694	20148
eilA101_50	50	50	4	4	96.3	*	843	—	96.3	*	843	364
eilA101_66	67	33	6	3	99.2		846	10913	99.6		846	434
eilA101_80	80	20	6	2	90.3	*	916	—	91.7	*	908	431
eilB101_50	50	50	7	7			n.a.		95.6	*	933	—
eilB101_66	67	33	9	5			n.a.		89.1	*	1056	293
eilB101_80	80	20	11	3			n.a.		97.2	*	1022	20199

*Optimality not proved.

Table 8.3. *Behavior of algorithm TV on the AVRPB instances of the third class. Computing times in Pentium* 60 *MHz seconds (time limit of* 6000 *seconds).*

Name	n	m	K	K_B	LB %		z^*	Time
FTV33_50	17	16	2	1	100.0		1841	3
FTV33_66	22	11	2	1	99.5		1899	123
FTV33_80	27	6	2	1	99.9		1704	37
FTV35_50	18	17	2	2	98.7		2077	367
FTV35_66	24	11	2	1	98.0		2150	653
FTV35_80	28	7	2	1	98.7		1996	412
FTV38_50	19	19	2	2	100.0		2162	56
FTV38_66	26	12	2	2	98.3		2132	1382
FTV38_80	31	7	3	1	98.8		1982	1303
FTV44_50	22	22	2	2	98.6		2348	1557
FTV44_66	30	14	2	1	97.2		2225	4035
FTV44_80	36	8	3	1	98.0		2184	3439
FTV47_50	24	23	2	2	99.6		2343	230
FTV47_66	32	15	2	1	99.7		2427	288
FTV47_80	38	9	2	1	99.3		2312	1950
FTV55_50	28	27	2	2	99.2		2425	2450
FTV55_66	37	18	2	1	98.6		2246	5045
FTV55_80	44	11	2	1	98.6		2264	5091
FTV64_50	32	32	2	2	98.7		2728	4635
FTV64_66	43	21	2	1	98.5		2673	5797
FTV64_80	52	12	3	1	98.0	*	2679	—
FTV70_50	35	35	2	2	98.3	*	2940	—
FTV70_66	47	23	2	1	99.6		2808	4950
FTV70_80	56	14	2	1	99.3		2688	5049

*Optimality not proved.

and K_B are determined by solving the associated BPP using the code MTP by Martello and Toth [26]. The tables also report for each algorithm

- the percentage error of the lower bound, LB, computed at the root node;
- the value of the best solution found by the algorithm, z^*; if the optimality of z^* is not proved (either because the time limit has been reached or because EHP was not able to check the optimality of $v(\mathrm{P2}')$), z^* indicates the best incumbent solution value and the instance is marked with an asterisk;
- the overall computing time expressed in CPU seconds.

Percentage errors are computed as the ratio of the lower bound divided by the best z^* (i.e., the optimal or the best overall incumbent solution value) and multiplied by 100.

The performance of algorithms TV and EHP can be compared only on the 64 VRPB instances (34 of the first class and 30 of the second class) considered by both algorithms. Tables 8.1 and 8.2 show that the lower bound at the root node obtained by EHP is generally

better than that obtained by TV (of 64 instances, the lower bound of TV gives a superior value in only 3 cases). As for the overall algorithms, it is difficult to compare directly the corresponding computing times, since they refer to different machines. Also, with respect to the number of instances for which the algorithms were not able to prove the optimality of the solution found, a comparison is difficult, because of both the different speeds of the machines and the different time limit imposed on the execution of each instance. Of the 64 common instances, TV and EHP were not able to prove the optimality of the solution in 12 and 11 cases, respectively. In addition, it has to be noted that for 4 of the 12 instances unsolved by TV, algorithm EHP was able to find better feasible solutions (instances G1, G4, eilD73_80, and eilA101_80).

As pointed out by Mingozzi, Giorgi, and Baldacci [28], for algorithm EHP it is better to have only a few linehaul and backhaul customers per route (say, an average of about 10 customers of each type, as occurs for the instances of the first two classes). In fact, in this case the sizes of $\mathcal{L}'$ and $\mathcal{B}'$ are relatively small, and problem P2$'$ can be solved in reasonable computing times.

As for the AVRPB instances of the third class, it can be noted that the additive lower bound of algorithm TV is on average tighter on these instances than for those of the first two classes.

8.5 Heuristic Algorithms

Several heuristic algorithms proposed for the solution of VRPB are extensions of algorithms for CVRP (see Chapters 5 and 6). Most of these heuristics were developed for the symmetric, and in many cases only the Euclidean, version of the problem. The extension of these methods to AVRPB is often difficult, or even impossible, and may lead to unpredictable results.

8.5.1 Algorithm of Deif and Bodin

The first heuristic algorithm for VRPB was proposed by Deif and Bodin [9]. The algorithm, called DB herein, is an extension of the well-known Clarke and Wright [8] heuristic for CVRP. The Clarke and Wright algorithm starts with an infeasible solution where each customer is visited by a separate route. Routes are then iteratively combined by considering the saving, in terms of routing cost, that can be achieved by serving two customers on the same route instead of leaving them on separate routes. The saving obtained by visiting customers i and j in sequence on the same route can be expressed as

$$s_{ij} = c_{0i} + c_{i0} + c_{0j} + c_{j0} - (c_{0i} + c_{ij} + c_{j0}) = c_{i0} + c_{0j} - c_{ij}. \tag{8.48}$$

The results obtained by using the standard Clarke and Wright algorithm for the solution of VRPB are greatly affected by the fact that the precedence constraint substantially reduces the number of feasible merging. Deif and Bodin experimentally showed that for VRPB the best results are obtained by delaying the formation of mixed routes. Hence, they proposed modifying the saving definition to penalize the arcs connecting customers of different types,

thus delaying the union of linehaul and backhaul routes. The *backhaul saving* is defined as

$$s'_{ij} = \begin{cases} s_{ij} - pS & \text{if } i \in L,\ j \in B \text{ or vice versa,} \\ s_{ij} & \text{otherwise,} \end{cases} \tag{8.49}$$

where S is an estimate of the maximum saving s_{ij} and p is a real penalty between 0 and 1.

The Clarke and Wright method, and hence algorithm DB, does not allow for the control of the number of routes of the final solution. Indeed, the solution found for a given instance can require more than K routes to serve all the vertices, thus being infeasible. From a practical point of view, both the routing cost of the solution obtained with algorithm DB and the probability that this solution is feasible are strongly related to the number of route mergings executed. It is then evident that, even if delayed, the route orientation arising in the mixed routes, and the consequent decrease of possible route merging combinations, reduces the effectiveness of this method in facing the VRPB in terms of both the overall routing cost and the number of feasible solutions found. It can be noted that algorithm DB may be easily adapted to consider AVRPB instances (see, e.g., Vigo [35] for a discussion on the extension of the Clarke and Wright method to the asymmetric CVRP).

Deif and Bodin [9] tested their algorithm on randomly generated problem instances with 100 to 300 customers and a backhaul percentage between 10% and 50%. Several p values were tested, and the results obtained show that values of p between 0.05 and 0.20 produced the best solutions.

8.5.2 Algorithms of Goetschalckx and Jacobs-Blecha

Goetschalckx and Jacobs-Blecha [19] proposed an algorithm, called SF herein, for the VRPB with Euclidean cost matrix. The approach is based on the concept of space-filling curves, previously used by Bartholdi and Platzman [3] for the solution of the planar TSP. Using the space-filling curve transformation, linehaul and backhaul customers are, separately, transformed from points in the plane into points along a line. The two separate sequences of points are then partitioned to form feasible routes each containing customers of only one type. Each linehaul route is, in turn, merged with the nearest backhaul route (according to the space-filling mapping), thus obtaining the final set of vehicle routes. Also in this case, the method does not guarantee building solutions using exactly K routes. Goetschalckx and Jacobs-Blecha tested both the DB and SF algorithms on 57 Euclidean instances with 25 to 150 vertices, 20% to 50% of which are backhauls. The results presented by Goetschalckx and Jacobs-Blecha [19] show that DB solutions are generally better than those obtained by SF, while SF is faster than DB, mainly for large problems.

More recently, Goetschalckx and Jacobs-Blecha [20] presented a heuristic algorithm, called LHBH, for the Euclidean version of VRPB. The approach is based on the extension of the generalized assignment heuristic proposed by Fisher and Jaikumar [15] for CVRP. Initially, a partial solution made up of K *route primitives* is obtained as follows: first, K *seed radials* are determined by iteratively solving a capacitated location-allocation problem; then for each such radial a route primitive is obtained by considering the customers located close to the radial (i.e., within a 10-degree angle) and sequencing the linehaul customers by increasing distance to the depot and the backhaul customers by decreasing distance. The customers are then grouped together into K clusters by heuristically solving generalized assignment problems, whose cost matrices contain the insertion cost of every vertex into each

of the route primitives previously determined. Finally, the routes are determined through a modified insertion heuristic algorithm for TSP and postoptimization exchange procedures. Heuristic LHBH was tested on the Euclidean problems described in Goetschalckx and Jacobs-Blecha [19] (i.e., the instances of the first class illustrated in section 8.1.1), and it obtained better results than the previously proposed algorithms.

8.5.3 Algorithm of Toth and Vigo

In this section we describe the cluster-first, route-second heuristic algorithm, called HTV, proposed by Toth and Vigo [34] for both VRPB and AVRPB. The heuristic uses a general clustering method that exploits the information of the solutions associated with a lower bound. This is motivated by the fact that the solutions obtained through good relaxations of an optimization problem often are almost feasible and contain a high degree of information on the optimal solution structure. Therefore, relaxation-based clustering may provide a good starting point for heuristics based on local search, which can obtain feasible solutions quickly. For other examples on the use of relaxations to initialize heuristic algorithms for routing problems, see, e.g., Fisher [14] for the symmetric CVRP and Vigo [35] for the asymmetric CVRP.

The relaxation used to initialize algorithm HTV is the Lagrangian approach described in section 8.3.3. In the following we describe algorithm HTV by examining its main steps separately.

Clustering Step. The first step of algorithm HTV requires the construction of groups of customers, called *clusters*, containing only linehaul or backhaul customers. These clusters are later combined to form a basic set of, possibly infeasible, routes. The Lagrangian solution (i.e., the optimal solution to the current Lagrangian problem) is used to define K linehaul clusters and $\overline{K}$ backhaul clusters, where $\overline{K}$, with $K_B \leq \overline{K} \leq K_M = \min\{K, m\}$, is the number of arcs connecting the backhaul customer set to the depot in the Lagrangian solution (see section 8.3.2). To exploit the information on the optimal solution structure embedded in the Lagrangian solution, the connected components obtained by removing the $2 \cdot K$ arcs incident in the depot and the $\overline{K}$ interface arcs, connecting linehaul to backhaul customers, are chosen as clusters. Thus, given the solution of the relaxed problem, the initial clusters can be determined very simply in $O(n + m)$ time. Note that some of the clusters obtained at the end of this phase can be infeasible with respect to the capacity constraint.

Matching-Routing Step. The clusters are then combined to define the subsets of customers associated with the initial routes. In particular, $\overline{K}$ linehaul clusters must be associated with the $\overline{K}$ backhaul clusters, hence forming mixed routes, while the remaining $K - \overline{K}$ linehaul clusters, if any, are associated with the depot. To this end, the removed interface arcs of the Lagrangian solution are first considered. If a feasible (with respect to the capacity constraint) linehaul cluster is connected to one or more feasible backhaul clusters, one of them is arbitrarily chosen and combined with the linehaul cluster. The remaining clusters, say, $K' \leq K$ and $\overline{K}' \leq \overline{K}$, are then combined by solving an associated assignment problem. Let us define a $K' \times K'$ matrix, γ, whose rows are associated with the linehaul clusters. The first $\overline{K}'$ columns are associated with the backhaul clusters and the remaining $(K' - \overline{K}')$ columns with the depot. The value of γ_{pq} is an estimation of the

cost incurred by serving on the same route the linehaul customers of cluster p and either the backhaul customers of cluster q (if $q \leq \overline{K}'$) or the depot (if $q > \overline{K}'$). This value is computed as the cost of a heuristic solution to the TSPB associated with clusters p, q (if any) and the depot. For each customer subset, the corresponding initial, possibly infeasible, route is built by using a farthest-insertion TSP heuristic, modified to take into account the precedence constraint between linehaul and backhaul customers.

Intraroute Postoptimization. Each route is improved by applying a postoptimization procedure that considers all the feasible (with respect to the precedence constraint between linehaul and backhaul customers) exchanges of two and three arcs belonging to the route (the so-called *intraroute two-exchanges* and *three-exchanges*). The procedure is similar to those described by Christofides and Eilon [6] and Kindervater and Savelsbergh [23]. The final solution is obtained by iteratively evaluating the cost of the route produced by each feasible exchange of two or three arcs, and by performing the best exchange among all those producing a positive cost reduction. The procedure is iterated until no cost reduction is found. As usual, only the two-exchanges are considered first, and then the three-exchange procedure is applied. For both the undirected and the directed case, the evaluation of the cost of the route produced by an exchange can be performed in constant time, through parametric labeling techniques. Hence, the overall time complexity of one iteration of the three-exchange procedure is $O(\overline{n}^3 + \overline{m}^3)$, where $\overline{n}$ and $\overline{m}$ are, respectively, the number of linehaul and backhaul customers of the route considered (see, e.g., Vigo [35] for the application of parametric labeling techniques to the asymmetric CVRP).

Interroute Postoptimization. The final set of routes is obtained by using local-search procedures based on the so-called *interroute one-exchanges* and *two-exchanges*. In other words, all the feasible movements of a customer from one route to another and all the feasible exchanges of two arcs belonging to different routes are evaluated. Exchanges that increase the infeasibility of an infeasible route or that produce an infeasible route starting from feasible ones are not considered. The score of an exchange is the weighted sum of the routing saving and, if one of the two routes involved in the exchange is infeasible, of the overload reduction produced by the exchange. For each procedure the final solution is obtained by iteratively evaluating the score produced by each exchange and by performing the best exchange until no improvement is found. At each iteration, $O(n^2 + m^2)$ exchanges are considered. The feasibility check and the computation of the score for each exchange can be executed in constant time through parametric labeling techniques. Hence, the computational complexity of a single iteration is $O(n^2 + m^2)$. The intraroute postoptimization procedure is then applied to each final route.

8.5.4 Computational Results for the Heuristics

The heuristic algorithm HTV described in section 8.5.3 was implemented in FORTRAN and run on an IBM 486/33 personal computer for the three classes of VRPB and AVRPB test problems illustrated in section 8.1.1. Tables 8.4, 8.5, and 8.6 give the results obtained by HTV, compared with the value of the optimal solution or of the best available lower bound, and, where possible, with the results obtained by algorithms DB, SF, and LHBH.

Table 8.4. *Behavior of the heuristic algorithms on the VRPB instances of the first class. Computing times in IBM* 386/20 *seconds.*

						DB		SF		LHBH		HTV		
Name	n	m	K	K_L	K_B	%ratio	Time	%ratio	Time	%ratio	Time	%ratio	Time	%r. Best
A1	20	5	8	7	2	100.4	0.7	117.0	6.6	100.3	7.8	100.0	24.1	100.0
A2	20	5	5	4	1	102.8	0.6	119.6	5.6	100.6	7.8	100.0	16.7	100.0
A3	20	5	4	3	1	103.6	0.5	111.7	5.4	104.0	7.8	100.0	7.3	100.0
A4	20	5	3	3	1	105.7	0.6	126.8	4.9	101.7	7.8	100.0	8.0	100.0
B1	20	10	7	7	4	106.3	0.9	115.8	7.7	100.3	5.7	100.0	47.7	100.0
B2	20	10	5	4	3	106.8	0.7	124.3	5.8	100.2	5.7	100.0	25.3	100.0
B3	20	10	3	3	2	101.7	0.7	121.6	4.9	100.0	5.7	100.0	0.8	100.0
C1	20	20	7	6	6	106.4	1.6	126.5	8.1	102.7	18.0	100.4	80.3	100.4
C2	20	20	5	4	4	100.0	1.4	116.9	6.6	103.2	18.0	100.0	43.2	100.0
C3	20	20	5	3	3	102.3	1.4	—	—	102.2	18.0	100.0	16.2	100.0
C4	20	20	4	3	3	105.4	1.8	116.7	6.8	102.3	18.0	100.0	19.4	100.0
D1	30	8	12	10	3	104.4	2.0	110.8	9.6	100.6	9.8	100.1	73.1	100.0
D2	30	8	11	10	3	106.3	2.5	—	—	101.4	9.8	100.0	80.2	100.0
D3	30	8	7	6	2	101.2	2.0	110.7	7.2	100.1	9.8	100.0	55.6	100.0
D4	30	8	5	5	2	101.4	1.9	109.5	6.6	101.1	9.8	100.0	41.2	100.0
E1	30	15	7	6	3	102.3	2.4	123.9	8.0	102.3	17.7	100.0	88.2	100.0
E2	30	15	4	4	2	102.9	2.5	132.1	7.4	105.4	17.7	100.3	53.2	100.0
E3	30	15	4	3	2	104.6	2.3	130.4	7.9	103.8	17.7	100.0	42.6	100.0
F1	30	30	6	5	6	110.9	4.9	—	—	105.3	25.2	100.0	120.3	100.0
F2	30	30	7	5	6	109.4	7.3	127.9	9.3	101.5	25.2	100.5	120.3	100.1
F3	30	30	5	4	4	107.1	7.1	125.0	9.1	101.7	25.2	100.2	90.1	100.7
F4	30	30	4	3	3	104.8	6.2	126.3	9.6	103.3	25.2	100.2	73.7	100.0
G1	45	12	10	9	3	101.9	7.5	122.1	9.1	103.2	32.7	100.3	120.7	100.3
G2	45	12	6	6	2	100.3	5.2	120.7	8.5	100.9	32.7	100.0	120.3	100.0
G3	45	12	5	5	2	103.7	6.4	120.4	9.6	100.8	32.7	100.8	109.9	100.4
G4	45	12	6	5	2	103.4	6.4	129.3	9.6	102.9	32.7	100.1	120.1	100.1
G5	45	12	5	4	1	105.4	6.0	128.6	9.4	102.7	32.7	100.7	109.0	100.6
G6	45	12	4	3	1	102.2	6.0	121.5	15.2	102.0	32.7	100.0	61.5	100.0
H1	45	23	6	6	3	107.0	7.6	126.6	10.2	105.6	32.5	102.5	140.4	102.4
H2	45	23	5	5	3	107.2	6.4	128.1	13.8	103.9	32.5	100.0	119.2	100.0
H3	45	23	4	4	2	106.5	6.2	124.4	13.3	103.6	32.5	100.0	111.6	100.0
H4	45	23	5	4	2	108.0	5.9	118.1	12.6	104.8	32.5	100.9	134.7	100.0
H5	45	23	4	3	2	104.7	6.4	112.7	13.9	104.9	32.5	100.0	68.1	100.0
H6	45	23	5	3	2	106.5	6.0	118.6	12.7	104.8	32.5	100.0	68.0	100.0
I1	45	45	10	8	9	104.9	13.2	127.8	13.5	104.1	42.4	103.5	181.9	103.5
I2	45	45	7	6	7	103.2	13.8	127.4	13.2	103.5	42.4	101.3	180.5	101.7
I3	45	45	5	4	5	107.2	25.2	117.3	32.0	106.9	42.4	104.0	180.4	103.0
I4	45	45	6	4	5	104.8	27.6	123.8	25.2	102.8	42.4	100.9	180.3	100.4
I5	45	45	7	4	5	101.2	12.7	123.0	19.0	101.3	42.4	101.1	180.4	100.5
J1	75	19	10	10	3	104.9	21.4	—	—	107.0	74.3	104.3	201.4	103.8
J2	75	19	8	8	2	108.1	25.0	121.2	16.1	105.2	74.3	106.7	200.4	105.8
J3	75	19	6	5	2	106.0	23.1	124.0	21.0	107.0	74.3	104.5	200.2	103.4
J4	75	19	7	7	2	107.8	23.6	120.2	18.4	106.1	74.3	106.0	200.3	104.6
K1	75	38	10	10	5	109.0	52.8	116.6	16.5	106.4	99.8	105.9	351.9	106.0
K2	75	38	8	8	4	110.5	67.3	121.4	17.0	104.1	99.8	104.4	352.4	103.8
K3	75	38	9	8	4	109.2	59.6	126.4	19.1	107.2	99.8	105.6	351.0	103.8
K4	75	38	7	7	3	109.3	77.0	123.0	20.3	105.1	99.8	106.1	351.6	105.7
L1	75	75	10	9	9	114.0	86.8	146.0	22.2	110.5	159.0	112.6	450.8	111.9
L2	75	75	8	8	8	120.5	158.6	137.6	27.3	108.3	159.0	111.9	450.8	108.6
L3	75	75	9	8	8	110.9	116.6	—	—	107.8	159.0	108.3	451.7	106.2
L4	75	75	7	6	6	112.7	122.2	132.3	107.2	107.6	159.0	107.0	450.9	108.0
L5	75	75	8	6	6	114.0	128.0	—	—	106.1	159.0	106.7	451.6	103.6
M1	100	25	11	10	3	120.8	105.7	137.7	24.0	114.5	157.3	114.0	381.2	113.3
M2	100	25	10	10	3	120.0	75.7	—	—	117.3	157.3	118.4	383.0	117.6
M3	100	25	9	9	3	115.5	65.7	131.4	28.5	114.8	157.3	112.9	382.8	112.1
M4	100	25	7	7	2	116.0	70.8	129.1	44.4	112.3	157.3	110.5	381.6	108.7
N1	100	50	11	10	5	119.5	148.2	156.8	5.0	113.5	159.0	111.7	453.8	111.9
N2	100	50	10	10	5	120.8	146.3	—	—	116.7	159.0	114.4	450.8	115.1
N3	100	50	9	9	4	116.1	146.1	134.6	27.6	109.5	159.0	112.3	452.1	111.8
N4	100	50	10	9	4	116.0	111.5	131.6	27.4	110.0	159.0	110.7	454.2	109.2
N5	100	50	7	7	3	118.3	145.8	126.1	40.4	110.4	159.0	109.5	452.3	109.5
N6	100	50	8	7	3	113.3	128.1	123.5	33.9	108.1	159.0	108.0	453.4	107.2
Average results						107.9	37.5	124.5	16.6	105.1	64.4	103.7	193.9	103.3

Table 8.5. *Behavior of the heuristic algorithms on the VRPB instances of the second class. Computing times in IBM* 486/33 *seconds.*

					DB			SF			HTV		
Name	n	m	K	K_B		%ratio	Time		%ratio	Time	%ratio	Time	%r. Best
eil22_50	11	10	3	2		115.6	0.1		132.6	1.1	100.0	5.1	100.0
eil22_66	14	7	3	1	(4)	115.8	0.2	(4)	134.7	1.2	100.0	4.8	100.0
eil22_80	17	4	3	1		100.0	0.2	(4)	140.3	1.1	100.0	7.0	100.0
eil23_50	11	11	2	1		103.8	0.2	(3)	114.1	1.3	100.0	2.6	100.0
eil23_66	15	7	2	1		109.6	0.2	(3)	123.6	1.2	100.0	5.5	100.0
eil23_80	18	4	2	2		111.6	0.2		109.3	1.3	100.0	3.9	100.0
eil30_50	15	14	2	2	(3)	115.2	0.2		145.3	1.8	100.0	3.3	100.0
eil30_66	20	9	3	1		110.6	0.2		149.3	1.7	100.4	7.4	100.0
eil30_80	24	5	3	1		114.0	0.2	(4)	151.4	1.8	101.6	7.5	100.6
eil33_50	16	16	3	2		120.9	0.2		128.2	1.9	100.5	16.4	100.1
eil33_66	22	10	3	1		116.8	0.2	(4)	154.3	1.8	100.4	15.8	100.0
eil33_80	26	6	3	1	(4)	121.8	0.3	(4)	142.2	1.9	100.1	15.9	100.1
eil51_50	25	25	3	3	(4)	119.7	0.7		124.5	2.1	100.5	40.8	100.5
eil51_66	34	16	4	2		117.0	0.7		119.7	2.3	100.9	48.5	100.0
eil51_80	40	10	4	1	(5)	115.9	1.9	(5)	124.4	2.1	101.6	53.1	101.6
eilA76_50	37	38	6	5		113.7	2.6		124.8	2.8	102.3	164.3	100.5
eilA76_66	50	25	7	4		118.1	2.7		137.6	2.7	101.6	148.3	101.6
eilA76_80	60	15	8	2	(9)	119.7	2.7	(9)	127.4	2.9	109.2	238.2	109.2
eilB76_50	37	38	8	7	(9)	119.5	2.8	(9)	145.1	3.1	103.0	240.5	103.0
eilB76_66	50	25	10	5		112.7	2.7	(11)	140.7	2.8	102.1	241.0	102.1
eilB76_80	60	15	12	3		112.3	3.0		125.7	3.5	103.2	240.7	103.0
eilC76_50	37	38	5	4		120.8	2.7		131.0	2.6	100.3	110.5	100.3
eilC76_66	50	25	6	3		116.2	3.0		131.6	3.2	101.5	148.7	101.1
eilC76_80	60	15	7	2		116.9	2.8		133.3	3.2	105.9	219.4	105.0
eilD76_50	37	38	4	3		116.8	2.8		124.6	3.9	100.1	93.7	100.0
eilD76_66	50	25	5	2		118.4	2.7		127.4	3.0	101.7	89.7	101.7
eilD76_80	60	15	6	2		116.2	2.7		128.7	3.7	103.3	190.6	103.3
eilA101_50	50	50	4	4	(5)	112.4	6.8	(5)	125.7	5.1	104.9	213.5	104.3
eilA101_66	67	33	6	3		113.3	6.4		133.5	5.6	103.0	240.6	102.8
eilA101_80	80	20	6	2	(7)	114.8	6.6	(7)	132.5	5.0	108.0	241.0	108.0
eilB101_50	50	50	7	7	(8)	118.5	7.0	(8)	136.1	5.6	107.8	241.6	106.7
eilB101_66	67	33	9	5	(10)	122.1	6.6	(10)	140.1	6.1	110.5	241.3	110.5
eilB101_80	80	20	11	3	(12)	118.2	7.6	(12)	134.1	5.8	106.7	241.6	106.6
Average results						115.4	2.4		132.5	2.9	102.5	114.6	102.2

For each instance the tables give the problem description and, for each heuristic algorithm, the following information:

(i) the percentage ratio of the solution with respect to the optimal solution value or to the best known lower bound value;

(ii) the computing time, expressed in IBM 386/20 seconds for Table 8.4 and in IBM 486/33 seconds for Tables 8.5 and 8.6.

Table 8.6. *Behavior of the heuristic algorithms on the AVRPB instances of the third class. Computing times in IBM* 486/33 *seconds.*

					DB		HTV		
Name	n	m	K	K_B	%ratio	Time	%ratio	Time	%r. Best
FTV33_50	17	16	2	1	131.9	0.1	100.0	1.4	100.0
FTV33_66	22	11	2	1	113.0	0.2	100.1	12.4	100.1
FTV33_80	27	6	2	1	124.9	0.2	100.0	11	100.0
FTV35_50	18	17	2	2	130.9	0.1	101.7	16.2	101.7
FTV35_66	24	11	2	1	125.4	0.2	101.9	16.7	101.9
FTV35_80	28	7	2	1	123.6	0.2	100.3	15.4	100.3
FTV38_50	19	19	2	2	114.6	0.2	100.0	20.2	100.0
FTV38_66	26	12	2	2	125.8	0.3	101.7	22.3	101.5
FTV38_80	31	7	3	1	131.2	0.3	100.4	40.7	100.0
FTV44_50	22	22	2	2	118.0	0.5	100.8	54.7	100.6
FTV44_66	30	14	2	1	123.3	0.5	100.4	26.8	100.4
FTV44_80	36	8	3	1	123.5	0.6	101.6	62.5	100.0
FTV47_50	24	23	2	2	119.7	0.9	100.4	37.2	100.4
FTV47_66	32	15	2	1	118.8	1.1	101.3	36.5	101.3
FTV47_80	38	9	2	1	118.5	1.1	101.9	32.2	101.9
FTV55_50	28	27	2	2	114.3	1.6	101.4	71.5	101.4
FTV55_66	37	18	2	1	120.1	1.7	103.4	49.9	102.5
FTV55_80	44	11	2	1	114.3	1.7	102.8	49.6	102.8
FTV64_50	32	32	2	2	135.7	2.2	102.2	116.9	102.2
FTV64_66	43	21	2	1	136.6	2.2	100.2	63.2	100.2
FTV64_80	52	12	3	1	117.9	2.4	101.9	93.4	101.9
FTV70_50	35	35	2	2	113.4	2.3	102.7	124	102.7
FTV70_66	47	23	2	1	128.9	2.4	100.8	76.9	100.8
FTV70_80	56	14	2	1	125.3	2.6	100.7	58.1	100.7
Average results					121.6	1.1	101.2	46.2	101.0

Note that the percentage ratio computed by using the lower bound value is an upper bound on the percentage ratio of the heuristic solution value with respect to the optimal solution value.

Algorithm HTV is applied to the Lagrangian solution obtained at each subgradient iteration of the Lagrangian lower bounding procedure, until 200 iterations are executed or a prefixed time limit is reached. Each result reported in the tables is the best one obtained over all the iterations, and the computing time includes the lower bound computation. The "% r. Best" column in the tables gives the percentage ratio of the best solutions found by using different parameter settings or by allowing a longer computing time.

Table 8.4 reports the results obtained by algorithms DB, SF, LHBH, and HTV for the 62 VRPB instances of the first class. The value of the solution obtained by all the algorithms was computed by using a real-valued cost matrix and by rounding the final solution value to the nearest integer. (The solution values and the computing times (expressed in IBM 386/20 seconds) for algorithms DB, SF, and LHBH were kindly provided by Marc Goetschalckx and Charlotte Jacobs-Blecha.) The computing times of algorithm HTV reported in Table 8.4 were multiplied by four since, according to our experience with this kind of algorithm,

an IBM 486/33 is almost four times faster than the IBM 386/20 used by Goetschalckx and Jacobs-Blecha. Columns labeled HTV are obtained by imposing a time limit equal to $2(n + m)$ seconds for instances with $(n + m) < 100$ and equal to $3(n + m)$ seconds otherwise. Table 8.4 shows that algorithm HTV performed better than the other algorithms from the literature (obtaining the best solution in 52 of the 62 test problems) within acceptable computing times. The greater effectiveness of algorithm HTV can also be seen by noting that, over all the instances of the first class, the average percentage ratio of the solutions obtained by this algorithm is 103.7%, whereas the solutions obtained by DB, SF, and LHBH have an average percentage ratio equal to 107.9%, 124.5%, and 105.1%, respectively. All the algorithms run in a reasonable computing time. Indeed, over all the instances of the first class, the average computing time needed by algorithm HTV is 193.9 seconds, while DB, SF, and LHBH require on average 37.5, 16.6, and 64.4 seconds, respectively.

The results for the 33 VRPB instances of the second class are illustrated in Table 8.5. The implementations of heuristics DB and SF, as coded by Toth and Vigo, were used in the comparison. A time limit of 240 seconds was imposed for algorithm HTV. The results confirm the good performance of algorithm HTV. Indeed, over all the instances of the second class, HTV always found the best solution, and the average percentage ratio of the solutions obtained by this algorithm is 102.5%, whereas the solutions obtained by DB and SF have an average percentage ratio equal to 115.4% and 132.5%, respectively. Moreover, algorithms DB and SF were unable to determine a solution with the desired number of routes in 12 and 16 problems, respectively. (The number of determined routes, when different from $K = K_L$, is reported in brackets.)

Table 8.6 shows the results obtained by heuristics HTV and DB on the 24 AVRPB instances of the third class. The results of heuristic DB (which may be easily modified to consider asymmetric instances) correspond to the implementation coded by Toth and Vigo. Also, in this case algorithm HTV obtains very good results: the average percentage ratio of the solutions obtained by this algorithm is 101.2%, whereas the solutions obtained by DB have an average percentage ratio equal to 121.6%.

Acknowledgments

This work was supported by Ministero dell'Università e della Ricerca Scientifica e Tecnologica and by Consiglio Nazionale delle Ricerche, Italy.

Bibliography

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] S. Anily. The vehicle-routing problem with delivery and back-haul options. *Naval Research Logistic Quarterly*, 43:415–434, 1996.

[3] J. Bartholdi and L. Platzman. An $O(n \log n)$ planar traveling salesman heuristic based on spacefilling curves. *Operations Research Letters*, 1:121–125, 1982.

[4] L. Bianco, A. Mingozzi, and S. Ricciardelli. A set partitioning approach to the multiple depot vehicle scheduling problem. *Optimization Methods and Software*, 3:163–194, 1994.

[5] D. Casco, B.L. Golden, and E.A. Wasil. Vehicle routing with backhauls: Models, algorithms, and case studies. In B.L. Golden and A.A. Assad, editors, *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam, 1988, pp. 127–147.

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

[7] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem based on the spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981.

[8] G. Clarke and J.V. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[9] I. Deif and L.D. Bodin. Extension of the clarke and wright algorithm for solving the vehicle routing problem with backhauling. In A. Kidder, editor, *Proceedings of the Babson College Conference on Software Uses in Transportation and Logistic Management*, Babson Park, MA, 1984, pp. 75–96.

[10] J.J. Dongarra. Performance of various computers using standard linear equations software. Technical Report CS–89–85, University of Tennessee, Knoxville, 1998.

[11] C. Duhamel, J.-Y. Potvin, and J.-M. Rousseau. A tabu search heuristic for the vehicle routing problem with backhauls and time windows. *Transportation Science*, 31:49–59, 1997.

[12] M. Fischetti and P. Toth. An additive bounding procedure for combinatorial optimization problems. *Operations Research*, 37:319–328, 1989.

[13] M. Fischetti, P. Toth, and D. Vigo. A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research*, 42:846–859, 1994.

[14] M.L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42:626–642, 1994.

[15] M.L. Fisher and R. Jaikumar. A generalized assignment heuristic for the vehicle routing problem. *Networks*, 11:109–124, 1981.

[16] H.N. Gabow and R.E. Tarjan. Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms*, 5:80–131, 1984.

[17] S. Gélinas, M. Desrochers, J. Desrosiers, and M.M. Solomon. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61:91–109, 1995.

[18] F. Glover and D. Klingman. Degree constrained spanning trees. *Colloquia Mathematica Societatis Janos Bolyai* 12, 1975, pp. 425–439.

[19] M. Goetschalckx and C. Jacobs-Blecha. The vehicle routing problem with backhauls. *European Journal of Operational Research*, 42:39–51, 1989.

[20] M. Goetschalckx and C. Jacobs-Blecha. The vehicle routing problem with backhauls: Properties and solution algorithms. Technical Report MHRC-TR-88-13, Georgia Institute of Technology, Atlanta, 1993.

[21] B.L. Golden, E. Baker, J. Alfaro, and Schaffer J. The vehicle routing problem with backhauling: Two approaches. In R. Hammesfahr, editor, *Proceedings of the* XXI *Annual Meeting of S.E. TIMS*, Myrtle Beach, SC, 1985, pp. 90–92.

[22] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations. *Annals of Operations Research*, 61:21–43, 1995.

[23] G.A.P. Kindervater and M.W.P. Savelsbergh. Vehicle routing: Handling edge exchanges. In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, Wiley, Chichester, UK, 1997, pp. 337–360.

[24] G. Kontoravdis and J.F. Bard. A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7:10–23, 1995.

[25] K. Malik and G. Yu. A branch and bound algorithm for the capacitated minimum spanning tree problem. *Networks*, 23:525–532, 1993.

[26] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Chichester, UK, 1990.

[27] A. Mingozzi, M.A. Boschetti, S. Ricciardelli, and L. Bianco. A set partitioning approach to the crew scheduling problem. *Operations Research*, 47:873–888, 1999.

[28] A. Mingozzi, S. Giorgi, and R. Baldacci. An exact method for the vehicle routing problem with backhauls. *Transportation Science*, 33:315–329, 1999.

[29] S. Salhi and G. Nagy. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of Operational Research Society*, 50:1034–1042, 1999.

[30] S.R. Thangiah, J.-Y. Potvin, and T. Sun. Heuristic approaches to vehicle routing with backhauls and time windows. *Computers and Operations Research*, 23:1043–1057, 1996.

[31] P. Toth and D. Vigo. An exact algorithm for the capacitated shortest spanning arborescence. *Annals of Operations Research*, 61:121–142, 1995.

[32] P. Toth and D. Vigo. A heuristic algorithm for the vehicle routing problem with backhauls. In L. Bianco and P. Toth, editors, *Advanced Methods in Transportation Analysis*, Springer-Verlag, Berlin, 1996, pp. 585–608.

[33] P. Toth and D. Vigo. An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31:372–385, 1997.

[34] P. Toth and D. Vigo. A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research*, 113:528–543, 1999.

[35] D. Vigo. A heuristic algorithm for the asymmetric capacitated vehicle routing problem. *European Journal of Operational Research*, 89:108–126, 1996.

[36] C. Yano, T. Chan, L. Richter, L. Cutler, K. Murty, and D. McGettigan. Vehicle routing at quality stores. *Interfaces*, 17:52–63, 1987.