A GRASP for a Multi-depot Multi-commodity Pickup and Delivery Problem with Time Windows and Heterogeneous Fleet in the Bottled Beverage Industry

Roger Z. Ríos-Mercado[1], J. Fabián López-Pérez[2], and Andrés Castrillón-Escobar[1]

[1] Universidad Autónoma de Nuevo León (UANL), Graduate Program in Systems Engineering, AP 111–F, Cd. Universitaria, San Nicolás de los Garza, NL 66450, Mexico
{roger,andres}@yalma.fime.uanl.mx

[2] Universidad Autónoma de Nuevo León (UANL), Graduate Program in Management Science, CEDEEM, San Nicolás de los Garza, NL 66450, Mexico
jesus.lopezpz@uanl.edu.mx

Abstract. A pickup and delivery vehicle routing problem from a real-world bottled-beverage distribution company is addressed. The problem consists of deciding how to load every trailer, how to configure the vehicles in terms of the trailers, and how to route the vehicles, so as to minimize routing and fixed costs. The problem includes several features such as time windows on both customers and vehicles, multiple depots, multiple product delivery, split delivery, heterogeneous fleet, and dock capacity, to name a few. To solve this problem a GRASP-based heuristic is proposed. Problem decomposition, vehicle composition and route construction mechanisms, and local search procedures for different types of neighborhoods are developed. The heuristic is empirically assessed over a wide set of instances. Empirical results show that the proposed method obtains solutions of better quality than those reported by the company.

Keywords: Vehicle routing, Pickup and delivery, Multi-commodity, Heterogeneous fleet, Metaheuristics, GRASP.

1 Introduction

The problem addressed in this paper comes from a bottled beverage distribution company located in the city of Monterrey, Mexico. This company needs to distribute its products across several distribution centers. The distribution centers can also in turn request to withdraw products or relocate products to other centers. Due to its specific features, this problem is classified as a vehicle routing problem with pickup and delivery (PDP) with multiple depots.

The problem consists of transporting products among plants and distribution centers in a network. In each plant, there is a determined number of vehicles that

D. Pacino, S. Voß, and R.M. Jensen (Eds.): ICCL 2013, LNCS 8197, pp. 143–157, 2013.

can be used to do the routing. In addition, a decision has to be made so as to haul one or two trailers into a vehicle. This decision has two main effects. As it turns out, both the cost and time of traveling between two given points depend on whether the vehicle is single or double. A single vehicle travels faster and costs less than a double vehicle. However, a double vehicle has more capacity and can carry a larger amount of product between points. There are time windows requirements in both distribution centers and vehicles. The later is due to the fact that a daily maintenance for each vehicle must be performed at specific times. The orders can be split into several deliveries and vehicles can also visit several times a distribution center if necessary. The planning horizon is daily. In addition, there is a dock capacity in each distribution center that must be met. Each vehicle must return to its original depot by the end of the day.

Each trailer has two compartments: one on the top and one on the bottom. For each trailer, the capacity of both top and bottom is known, but not necessarily equal for every trailer. Moreover, some trailers may have a fixed shelf (or division between the top and bottom compartments) which would become an important issue when loading the products. There are many different products handled by the company; however, by and large they can be divided into two main types of products: returnable (type R) and non-returnable (type N). This distinction is important because type R products are heavier. As a direct consequence of this, the following loading rules must be met: (i) if the trailer has a shelf, any product, regardless its type, can be placed in any part of the trailer compartments; (ii) if the trailer does not have a shelf, it is strictly forbidden to place a type R product on top of a type N product due to the weight difference between products. It is assumed that when a product of type R is delivered to a customer a matching amount of empty type R bottles is picked up. The goal is to find the best route configuration that minimizes the total cost associated with the product routing and vehicle usage. In doing so, decisions so as to what type of vehicle is to be used and how each trailer must be loaded must be made simultaneously.

Although the field of vehicle routing, particularly the class of PDPs, has been widely studied (e.g. [3, 6, 8–10]), to the best of our knowledge there is no other work in the literature addressing a PDP with all the features mentioned previously simultaneously present, namely, multiple depots, heterogeneous vehicles, time windows at nodes and vehicles, split deliveries, multiple products, vehicles with compartments, dock capacity. For recent extensive surveys on PDPs the reader is referred to the work of Berbeglia et al. [1], Drexl [4], and Parragh et al. [7].

In this paper, we introduce a mixed-integer linear programming (MILP) model for this problem. Given its inherent computational complexity, we propose a metaheuristic framework that combines decomposition, greedy randomized construction, and local search components. The effectiveness of the heuristic is empirically assessed. It was found that the proposed method found better routings and truck configurations than those reported by the company, resulting in important cost reductions.

2 Mathematical Formulation

For modeling the problem we define the following notation.
Sets

T Set of trailers.
T_i Subset of trailers that are initially located at node i.
$T^e(T^w)$ Subset of trailers with (without) a shelf.
K Set of vehicles.
K_i Subset of vehicles that are initially located at node i.
V Plants and distribution centers (nodes).
E Edges.
P Products.
$P_R(P_N)$ Subset of returnable (non-returnable) products.
H Time index (usually hours) set.

Parameters

c_t^{top} Capacity (number of pallets) on top compartment of trailer t.
c_t^{bot} Capacity (number of pallets) on bottom compartment of trailer t.
τ_t^l Time in which trailer t can start its route.
τ_t^u Time in which trailer t must return to its plant.
$\sigma(t)$ Original location of trailer t.
$\sigma(k)$ Original location of vehicle k.
n_{ip}^+ Amount of product p (number of 12-bottle boxes) that must be picked up from node i.
n_{ip}^- Amount of product p that must be delivered to node i.
(a_i, b_i) Time window for servicing node i.
o_p Number of 12-bottle boxes of product p that fit in a pallet.
c_{ij}^s Travel cost from node i to j using a single vehicle.
c_{ij}^d Travel cost from node i to j using a double vehicle.
s_{ij}^s Travel time from node i to j using a single vehicle.
s_{ij}^d Travel time from node i to$j$ using a double vehicle.
S Customer service time at each node.
f_{ih} Available dock capacity in node i at time h.
C Fixed cost for vehicle use.

Binary variables

w_{kt} = 1, if trailer t is assigned to vehicle k; 0, otherwise
x_{ijk} = 1, if vehicle k travels directly from node i to j; 0, otherwise
y_{ikh} = 1, if vehicle k arrives at node i at time h; 0, otherwise
z_k = 1, if vehicle k is used; 0, otherwise

Integer variables

v_k^{top} Number of pallets on top of vehicle k.
v_k^{bel} Number of pallets on bottom of vehicle k.

v_k^{type} Configuration of vehicle k (=-1/0/1 if unassigned/single/double).
g_{ikp}^- Amount of product p (number of 12-bottle boxes) that vehicle k will deliver at node i.
g_{ikp}^+ Amount of product p that vehicle k will pickup at node i.
q_{ikp} Amount of product p in vehicle k after servicing node i.

Note: At the start of a route, $g^-_{\sigma(k)kp} = 0$ and $q_{\sigma(k)kp} = g^+_{\sigma(k)kp}$, for $k \in K, p \in P$.

Real variables

(l_k, u_k) Starting and finishing time of vehicle k.
s_{ik} Starting service time of vehicle k at node i.
c_{ik} Accumulated cost of vehicle k up to node i.
OC_k Total travel cost for vehicle k.
OT_k Total travel time for vehicle k.

Model

$$\text{Minimize} \quad \sum_{k\in K} OC_k + \sum_{k\in K} Cz_k \tag{1}$$

$$\text{subject to} \quad \sum_{t\in T_{\sigma(k)}} w_{kt} \le 2 \qquad k \in K \tag{2}$$

$$\sum_{k\in K_{\sigma(t)}} w_{kt} \le 1 \qquad t \in T \tag{3}$$

$$v_k^{\text{type}} = \sum_{t\in T_{\sigma(k)}} w_{kt} - 1 \qquad k \in K \tag{4}$$

$$u_k \le \tau_t^u + M_T(1 - w_{kt}) \qquad k \in K, t \in T_{\sigma(k)} \tag{5}$$

$$l_k \ge \tau_t^l w_{kt} \qquad k \in K, t \in T_{\sigma(k)} \tag{6}$$

$$v_k^{\text{top}} \le \sum_{t\in T_{\sigma(k)}} c_t^{\text{top}} w_{kt} \qquad k \in K \tag{7}$$

$$v_k^{\text{bel}} \le \sum_{t\in T_{\sigma(k)}} c_t^{\text{bel}} w_{kt} \qquad k \in K \tag{8}$$

$$x_{ijk} \le z_k \qquad i, j \in V, k \in K \tag{9}$$

$$\sum_{t\in T_{\sigma(k)}} w_{kt} \ge z_k \qquad k \in K \tag{10}$$

$$\sum_{i:(i,j)\in E} x_{ijk} - \sum_{i:(j,i)\in E} x_{jik} = 0 \qquad j \in V, k \in K \tag{11}$$

$$\sum_{i:(i,j)\in E} x_{ijk} \le 1 \qquad j \in V, k \in K \tag{12}$$

$$a_i \sum_{j:(j,i)\in E} x_{jik} \le s_{ik} \le b_i \sum_{j:(j,i)\in E} x_{jik} \qquad i \in V, k \in K \tag{13}$$

$$a_{\sigma(k)} z_k \le OT_k \le b_{\sigma(k)} z_k \qquad k \in K \tag{14}$$

$$l_k - M_T(1 - z_k) \le s_{\sigma(k)k} \le u_k + M_T(1 - z_k) \qquad k \in K \tag{15}$$

$$l_k - M_T(1 - z_k) \leq OT_k \leq u_k + M_T(1 - z_k) \quad k \in K \tag{16}$$

$$\sum_{h \in H} h y_{ikh} \leq \frac{s_{ik}}{60} \leq \sum_{h \in H} (h+1) y_{ikh} \quad k \in K, i \in V \setminus \{\sigma(k)\} \tag{17}$$

$$\sum_{h \in H} h y_{\sigma(k)kh} \leq \frac{OT_k}{60} \leq \sum_{h \in H} (h+1) y_{\sigma(k)kh} \quad k \in K \tag{18}$$

$$\sum_{h \in H} y_{ikh} \leq 1 \quad i \in V, k \in K \tag{19}$$

$$\sum_{k \in K} y_{ikh} \leq f_{ih} \quad i \in V, h \in H \tag{20}$$

$$s_{ik} + S + s^s_{ij}\left(1 - v^{\text{type}}_k\right) + s^d_{ij} v^{\text{type}}_k - M_T(1 - x_{ijk}) \leq s_{jk} \quad k \in K, (i,j) \in E | j \neq \sigma(k) \tag{21}$$

$$s_{ik} + S + s^s_{i\sigma(k)}\left(1 - v^{\text{type}}_k\right) + s^d_{i\sigma(k)} v^{\text{type}}_k - M_T(1 - x_{i\sigma(k)k}) \leq OT_k \quad i \in V, k \in K \tag{22}$$

$$\sum_{k \in K} g^+_{ikp} \leq n^+_{ip} \quad i \in V, p \in P \tag{23}$$

$$\sum_{k \in K} g^-_{ikp} \geq n^-_{ip} \quad i \in V, p \in P \tag{24}$$

$$g^+_{ikp} \leq n^+_{ip} \sum_{j:(j,i) \in E} x_{jik} \quad i \in V, k \in K, p \in P \tag{25}$$

$$g^-_{ikp} \leq n^-_{ip} \sum_{j:(j,i) \in E} x_{jik} \quad i \in V, k \in K, p \in P \tag{26}$$

$$q_{ikp} + g^+_{jkp} - g^-_{jkp} - M_L(1 - x_{ijk}) \leq q_{jkp} \quad i \in V, k \in K, p \in P, j \in V \setminus \{\sigma(k)\} \tag{27}$$

$$q_{ikp} + g^+_{jkp} - g^-_{jkp} + M_L(1 - x_{ijk}) \geq q_{jkp} \quad i \in V, k \in K, p \in P, j \in V \setminus \{\sigma(k)\} \tag{28}$$

$$\sum_{p \in P} \frac{q_{ikp}}{o_p} \leq v^{\text{top}}_k + v^{\text{bel}}_k \quad i \in V, k \in K \tag{29}$$

$$\sum_{p \in P_N} \frac{q_{ikp}}{o_p} \leq v^{\text{bel}}_k + \sum_{t \in T_{\sigma(k)} \cap T^e} c^{\text{top}}_t w_{kt} \quad i \in V, k \in K \tag{30}$$

$$c_{ik} + c^s_{ij}\left(1 - v^{\text{type}}_k\right) + c^d_{ij} v^{\text{type}}_k - M_C(1 - x_{ijk}) \leq c_{jk} \quad i \in V, k \in K, j \in V \setminus \{\sigma(k)\} \tag{31}$$

$$c_{ik} + c^s_{i\sigma(k)}\left(1 - v^{\text{type}}_k\right) + c^d_{i\sigma(k)} v^{\text{type}}_k - M_C(1 - x_{i\sigma(k)k}) \leq c_{jk} \quad i \in V, k \in K \tag{32}$$

$$w_{kt} = 0 \quad k, t | \sigma(k) \neq \sigma(t) \tag{33}$$

$$g^-_{\sigma(k)kp} = 0 \quad k \in K, p \in P \tag{34}$$

$$q_{\sigma(k)kp} = g^{+}_{\sigma(k)kp} \qquad k \in K, p \in P \tag{35}$$

$$w_{kt}, y_{ikh}, z_k \in \{0,1\} \qquad i \in V, t \in T, k \in K, h \in H \tag{36}$$

$$x_{ijk} \in \{0,1\} \qquad (i,j) \in E, k \in K \tag{37}$$

$$v^{\text{type}}_k \in \{-1,0,1\} \qquad k \in K \tag{38}$$

$$v^{\text{top}}_k, v^{\text{bel}}_k, g^{-}_{ikp}, g^{+}_{ikp}, q_{ikp} \in Z^{+} \qquad i \in V, k \in K, p \in P \tag{39}$$

$$l_k, u_k, s_{ik}, c_{ik} \geq 0 \qquad i \in V, k \in K \tag{40}$$

Here M_L, M_T, and M_C are large enough big-M constants to make the corresponding constraints redundant. The objective is to minimize the sum of vehicle routing cost and vehicle usage cost (1). Constraints (2) limit the number of trailers that may be assigned to a vehicle. Constraints (3) ensure that a single trailer can be assigned to at most one vehicle. Constraints (4) are used to identify the type of vehicle configuration. Constraints (5)-(6) set the vehicle time windows. Constraints (7) and (8) set the vehicle capacity on top and bottom compartments, respectively. Constraints (9) and (10) set the relationship between the z_k and the x_{ijk} and w_{kt} variables, respectively. Flow balance is assured by (11). Constraints (12) ensure that a vehicle may visit a node at most once. Constraints (13)-(14) and (15)-(16) guarantee the time windows for both customers and vehicles, respectively. Constraints (17)-(19) set the correct relationship between the binary time index variables y_{ijk} and the time variables s_{ik} and OT_k. The dock capacity constraints are given by (20). Constraints (21) and (22) are used to ensure that the time variables are consistent with travel and service times. Product availability and demand are set by (23) and (24), respectively. Constraints (25) and (26) ensure that a vehicle can load or unload product at node i only if it visits that node. Constraints (27)-(28) establish the connection on vehicle load between nodes i and j when vehicle k indeed travels directly from i to j. Vehicle capacity is set by (29) and (30), the latter considering that type N products can be placed in the bottom compartment of any vehicle or in the top compartment of vehicles with a shelf. Constraints (31)-(32) ensure the consistency of the cost variables. Conditions (33) make it impossible to assign trailers to vehicles in different starting locations. Constraints (34) and (35) establish initial delivery and pickup conditions for a vehicle at the start of its route. The nature of the decision variables are set in (36)-(40).

The problem is NP-hard [2]. The largest instance we could solve exactly after several hours of computing time by CPLEX branch-and-bound algorithm was in the order of 6 nodes, 7 trailers, and 5 products. Given real-world instances are significantly larger, we develop a heuristic framework for this problem.

3 Proposed Heuristic

GRASP [5] is a multi-start metaheuristic that has been widely used for finding good quality solutions to many hard combinatorial optimization problems. It relies on greedy randomized constructions and local search.

The proposed method is depicted in Procedure 1. The problem involves many decisions at different levels. Thus, to make the problem more tractable a decomposition approach is taken. The main idea is first to estimate point-to-point requests (done in Step 5), and then, based on this information, to apply the iterative GRASP phases of construction (Step 7) and local search (Step 8). Each of these components is described next.

Procedure 1. GRASP($\Delta\beta$, α, *limit_iter*)

Input: $\Delta\beta$, step parameter for cost matrix; α, RCL quality parameter; *limit_iter*, number of iterations
Output: X_{best}, best solution found
1: $X_{\text{best}} \leftarrow \emptyset$; $f(X_{\text{best}}) \leftarrow +\infty$
2: *find_shortest_paths*(G, C^s, C^d, S^s, S^d)
3: $\beta \leftarrow 0$
4: **while** ($\beta \leq 1$) **do**
5: $R \leftarrow$ *solve_TP*(β)
6: **for** (*iter* = 1 to *limit_iter*) **do**
7: $X \leftarrow constructSolution(\alpha, R)$
8: $X \leftarrow localSearch(X)$
9: **if** (X is better than X_{best}) **then**
10: $X_{\text{best}} \leftarrow X$
11: **end if**
12: **end for**
13: $\beta \leftarrow \beta + \Delta\beta$
14: **end while**
15: **return** X_{best}

Preprocessing: Construction of the optimal cost and time matrices: Initially we have an undirected graph where the edges handle two types of costs (c^s_{ij} and c^d_{ij}) and two types of times (s^s_{ij} and s^d_{ij}), for single and double vehicles. In this preprocessing phase (Step 2 of Procedure 1), we find optimal shortest paths between all pairs of nodes for each of the four matrices by applying the well-known Floyd-Warshall algorithm. Let $c^s_{[ij]}$ and $c^d_{[ij]}$ be the cheapest cost of traveling from node i to node j using a single and double vehicle, respectively. Similarly, let $s^s_{[ij]}$ and $s^d_{[ij]}$ represent the shortest time of traveling from i to j for a single and a double vehicle, respectively. This information on optimal paths is used in other components of the algorithm and needs to be computed only once.

Decomposition: Point-to-point request generation: As mentioned before, to make the problem more tractable we first attempt to estimate point-to-point requests. Each request or order is identified by a vector (i, j, p, r_{ijp}), where r_{ijp} is the amount of product p (measured in number of 12-bottle boxes) to be picked up at i and delivered to j. To compute these requests, we solve a transportation problem, where we take as input the information on pickup and delivery quantities at every node (given by parameters n^+_{ip} and n^-_{ip}, respectively), and the "cost" between nodes i and j. Now, we must bear in mind that this cost depends on whether

single or double vehicles are used which is unknown at this point. However, we can construct a cost function as a convex combination of the optimal costs for single and double vehicles parameterized by a weight $\beta \in [0,1]$. The output, for a fixed value of β is a set of requests R.

Let y_{ij} a binary-decision variable equal to 1 if at least one request from i to j exists and 0 otherwise. Then, the transportation problem TP(β) given below is solved in Step 5 of Procedure 1. Here, $M = \max_{i \in V, p \in P}\{n_{ip}^+\}$ is a large enough constant to make the constraint (44) redundant when $y_{ij} = 1$. The problem is not quite a classical transportation problem. Here we have a fixed cost rather than a variable cost. Also, we have several capacity/demand constraints for the different products. However, time is not an issue since this problem is easy to solve even for large instances. Note that different values of β give rise to different cost structure among nodes, and therefore, we proceed to generate different TPs for a variety of values of β. As can be seen in Procedure 1 we use a step size $\Delta\beta$ that allows us to discretize the range for β and try out different matrices and have therefore more diversity.

Model TP(β)

$$\text{Minimize} \quad f(r,y) = \sum_{i,j \in V} \left(\beta c^s_{[ij]} + (1-\beta) c^d_{[ij]}\right) y_{ij} \tag{41}$$

$$\text{subject to} \quad \sum_{j \in V} r_{ijp} \le n^+_{ip} \qquad i \in V, p \in P \tag{42}$$

$$\sum_{i \in V} r_{ijp} \ge n^-_{jp} \qquad j \in V, p \in P \tag{43}$$

$$\sum_{p \in P} r_{ijp} \le M y_{ij} \qquad i,j \in V \tag{44}$$

$$r_{ijp} \ge 0 \qquad i,j \in V, p \in P \tag{45}$$

$$y_{ij} \in \{0,1\} \qquad i,j \in V \tag{46}$$

Construction phase: Given a set of requests R, this phase is where these requests are assigned to vehicles, and, as a consequence, where routes are determined for these vehicles. The construction procedure is depicted in Procedure 2.

The procedure starts by initializing the set of vehicle routes $X_\pi = \{\pi_k | k \in K\}$ as the empty set (Step 1) and by assigning each trailer $t \in T$ to a vehicle $k \in K_{\sigma(t)}$ (Step 2), that is, initally we have single vehicles. In a later stage we consider merging two single vehicles into one double vehicle. Then a set R' containing all possible point-to-point pairs that need to be served and a set $\bar{P}$ containing all feasible routes are formed. Here, the latter is defined as $\bar{P} = \{(i,j,k) : (i,j) \in R' \wedge (i,j,k) \text{ is a feasible route}\}$, where the term feasibility means that vehicle k can deliver the products from node i to j within the vehicle and node time windows. Let P_k be the set of all feasible routes associated with vehicle k. Then, the actual route construction takes place within the *while* loop (Steps 5-11). Following the GRASP philosophy, a greedy function that measures the cost of assigning pair (i,j) to vehicle k is computed as follows:

Procedure 2. constructSolution(α, R)

```
1: X_π ← ∅
2: K ← T
3: R' = {⟨i, j⟩ : (i, j, p, r_ijp) ∈ R}
4: P̄ ← getFeasiblePaths(K, R')
5: while ( R ≠ ∅ ) do
6:       φ^min ← min{φ(i, j, k) : (i, j, k) ∈ P̄}
7:       φ^max ← max{φ(i, j, k) : (i, j, k) ∈ P̄}
8:       RCL = {(i, j, k) ∈ P̄ : φ(i, j, k) ≤ φ^min + α(φ^max − φ^min)}
9:       (i, j, k) ← chosen randomly from RCL
10:      (X_k, R) ← assignProducts(i, j, k, R, X)
11: end while
12: returnToDepot(X_Π)
13: return X
```

$$\varphi^{\delta}(i,j,k) = c^{\delta}_{[u(k)i]} + c^{\delta}_{[ij]} + c^{\delta}_{j\sigma(k)]}$$

where $\delta = s(d)$ if k is a single (double) vehicle. This function estimates the cost of traveling from the vehicle k current location $u(k)$ to node i then to j and then back to its depot $\sigma(k)$. Then, a restricted candidate list (RCL) is built (Step 9) with those elements whose greedy function value falls within α % of the best possible value. An element from RCL is randomly chosen. Step 10 performs the assignment of route (i, j) to vehicle k, and figures out the amount of product to be loaded by first assigning the non-returnable items without exceeding the non-returnable capacity, and then filling the vehicle with the returnable products, performing the necessary time updating for vehicle k and remaining orders to be served R. By proceeding this way, we guarantee that the product type priority rule (i.e., not placing type N products on top of type R in vehicles with no shelf) is met. Finally, in Step 12, we make sure that every vehicle returns from its current location to its depot.

Improvement Phase: A solution delivered by the construction phase might not necessarily satisfy the dock capacity constraints (20). Therefore, the goal of the local search is to attempt to improve the quality of the solution or to repair infeasibility if needed. The proposed improvement phase (depicted in Procedure 3) includes three different methods, which are described next.

Procedure 3. localSearch()

```
1: mergeVehicles()
2: transferShipment()
3: repairDocks()
4: return
```

Method 1: Merging single vehicles. The idea behind this method is to identify those pairs of single vehicles that use the same route and merge them

into a double vehicle (see Figure 1). This step is done only if feasibility with respect to the time window constraint is kept and the resulting cost of the merge is less than the sum of the cost of both single vehicles.

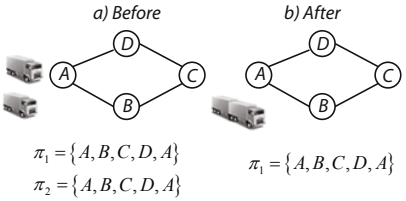

Fig. 1. Merging two single vehicles into one double vehicle

Method 2: Transfering loads between vehicles. The idea behind it is to analyze vehicles (either single or double) emanating from the same plant and try to combine both routes into a single route by one of the vehicles. This move is allowed only if the vehicles to be combined are of the same type and the resulting route is feasible with respect to the time window constraints. The method is depicted in Procedure 4, where u and v denote the vehicles to be combined with corresponding routes $\pi_u = \{u_1, u_2, \ldots, u_{\bar{u}}\}$ and $\pi_v = \{v_1, v_2, \ldots, v_{\bar{v}}\}$, respectively. Here, the neighborhood of possible moves is N_2 (Step 1).

Note in Steps 5-9, when combining these routes π_u and π_v, if vehicle u has still some load that must be delivered back to its depot $\sigma(u)$ or if vehicle v must pickup some load in $\sigma(k)$ then the resulting combined route would have to go through the depot resulting in $\pi_{u'} = \{u_1, \ldots, u_{\bar{u}}, v_2, \ldots, v_{\bar{v}}\}$; otherwise, the combined vehicle may skip the depot to have $\pi_{u'}$ as in Step 8.

Procedure 4. transferShipment()

1: $N_2 = \{(k_1, k_2) \in K \times K : \sigma(k_1) = \sigma(k_2) \wedge v_{k_1}^{\text{type}} = v_{k_2}^{\text{type}}\}$
2: **while** ($|N_2| > 0$) **do**
3: $(u, v) \leftarrow$ chose one element of N_2; $N_2 \leftarrow N_2 \setminus \{(u, v)\}$
4: $\pi_u = \langle u_1, \ldots, u_{\bar{u}} \rangle$; $\pi_v = \langle v_1, \ldots, v_{\bar{v}} \rangle$
5: **if** $\left(\sum_{p \in P} g^-_{\sigma(u)up} > 0 \vee \sum_{p \in P} g^+_{\sigma(v)vp} > 0 \right)$ **then**
6: $\pi_{u'} = \langle u_1, \ldots, u_{\bar{u}}, v_2, \ldots, v_{\bar{v}} \rangle$
7: **else**
8: $\pi_{u'} = \langle u_1, \ldots, u_{\bar{u}-1}, v_2, \ldots, v_{\bar{v}} \rangle$
9: **end if**
10: **if** ($\pi_{u'}$ is a feasible route path and $f(\pi_{u'}) < f(\pi_u) + f(\pi_v)$) **then**
11: $\pi_u \leftarrow \pi_{u'}$
12: $K \leftarrow K \setminus \{v\}$
13: $N_2 \leftarrow N_2 \setminus \{(k_1, k_2) : k_1 = v \vee k_2 = v\}$
14: **end if**
15: **end while**
16: **return**

Method 3: Repairing dock infeasibility. The solution delivered after applying the previous methods may not satisfy all dock capacity constraints. In that case, this method attempts to repair this possible infeasibility by making the necessary time shift adjustments. To perform this task, we define $J(h,i) = \{k \in K : \lfloor s_{ik}/60 \rfloor\}$ as the set of vehicles that arrive at node i during time index h (measured in hours). We now can determine the set of conflict nodes, that is, those nodes where the dock capacity constraints are violated at time h, as $\Psi(h) = \{i \in V : f_{ih} < |J(h,i)|\}$.

The method works as follows. Starting from the lowest value of h for which there are conflicts, we first choose one conflicting node i. Now, for this node we must determine what is the least critical vehicle, that is, the vehicle that possesses the largest amount of time adjustment flexibility. To do this, we compute for each vehicle the maximum amount of time that can be added to this vehicle without violating the future time window constraints in the remainder of its route $(i, \ldots, \bar{k})$ as follows

$$\Upsilon(i,k) = \min_{u \in (i,\ldots,\bar{k})} \{b_u - s_{uk}\}.$$

Then the vehicle with the largest possible value of $\Upsilon(i,k)$ is chosen and its future arrival times at every node in its remaining route are updated. By doing this, we guarantee that no more conflicting nodes for any time earlier than h arise, and this time adjustment may bring the current vehicle k at node i back into feasibility. The method is depicted in Procedure 5. If no vehicle can be found, the current iteration stops reporting an infeasible solution.

Procedure 5. repairDocks()

```
for h = 0, ..., |H| do
    while ( Ψ(h) ≠ ∅ ) do
        i ← choose one element of Ψ(h) arbitrarily
        while ( |J(h,i)| > f_ih ) do
            k* ← arg max_{k∈J(h,i)} {Υ(i,k)}
            for all (u ∈ [i, ..., ℓ_n]) do
                s_uk* = s_uk* + 60 − (s_ik* − 60 · ⌊s_ik*/60⌋)
            end for
        end while
    end while
end for
return
```

4 Empirical Results

The procedures were implemented in C# on Visual Studio 2010 environment. All experiments were performed on an Intel Core i7 computer running the Windows 7 operating system. CPLEX 12.1 was used for solving the transportation

subproblem. For each of the three data sets described below 20 instances were randomly generated based on distributions from real-world instances provided by the industrial partner.

Set A (small-size instances): 10 nodes, 20 trailers, 10 products.
Set B (medium-size instances): 15 nodes, 50 trailers, 40 products.
Set C (large-size instances): 25 nodes, 150 trailers, 300 products.

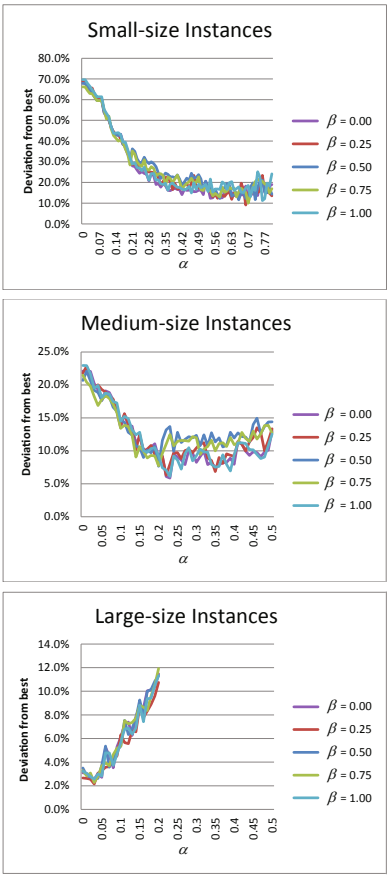

Fig. 2. Algorithm behavior as a function of α and β

Fine-Tuning of Parameters
The first experiment aims at finding appropriate values for algorithmic parameters α and β. To this end, we run a full two-factorial design. Parameter β was fixed at values in the [0.00, 1.00] range with increments of 0.25. Parameter α was fixed at values in the range [0.00, 0.80], [0.00, 0.50], and [0.00, 0.20] for the small-, medium-, and large-size instances, respectively, with increments of 0.01.

This results in 15,300 runs. The iteration limit was set at 1000 iterations for each run.

Figure 2 plots the results for the different values of α and β. The vertical axis indicate the average relative deviation from the best known feasible solution for each instance. One of the first observations we can make from the plot is that the parameter α has more influence over the solution quality than β does. This of course is a normal behavior in many GRASP implementations; however, we can also observe that the best possible value of this α depends on the instance size. For the smaller instances, the best solutions were obtained when α lies around 0.7. As the size of the instances grows, we can see how this best value of α goes down to around 0.20 for the medium-size instances, and 0.03 for the large-size instances. This means that reducing diversity in favor of more greedy-like solutions as instances get large provides a better strategy for finding solutions of better quality. For the remainder of the experiments we fixed α at 0.70, 0.20, and 0.03 for the small-, medium-, and large-size instances, respectively.

In the following experiment, we try to determine the effect that the step size $\Delta\beta$ has on algorithmic performance. While it is true that providing a finer discretization (i.e., reducing the step size $\Delta\beta$) could lead to more and better solutions, it is also true that the computational burden increases. Moreover, when the step size is sufficiently small, solutions obtained from different values of β will look alike, resulting in very little gain. Therefore, the purpose of this experiment is to investigate this effect. To this end, we run the heuristic fixing $\Delta\beta$ at different values as $\Delta\beta = 1/x$ with $x \in \{1, 2, \ldots, 10\}$. The iteration limit was set at 1000.

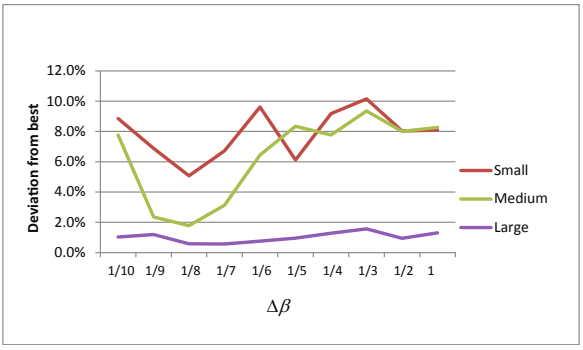

Fig. 3. Algorithm behavior as a function of $\Delta\beta$

Figure 3 displays the results plotting in the vertical axis the average deviation from the best known solution for every instance. The horizontal axis shows the different values of $\Delta\beta$. As can be seen from Figure 3, for the large-size instances the choice of $\Delta\beta$ did not have much impact; however, a slight improvement is observed at values around 1/8, which in fact matches the best value found for the small- and medium-size instances.

Feasibility Analysis
There is not guarantee that, in a given iteration of the algorithm the final solution delivered be feasible. Therefore it becomes important to assess the success rate of the heuristic in terms of its feasibility. To this end, we proceed to compute the percentage of times a feasible solution was obtained per iteration, and, given the algorithm returns the best solution found over all iterations, the percentage of times the algorithm delivered a feasible solution.

Table 1 shows the results for the different data sets. Columns 2-3 indicate the average and maximum success rate per iteration over all instances tested. The last column shows the success rate of the entire heuristic. As can be seen, even though some individual iterations may fail, it is empirically observed that the heuristic is always successful.

Table 1. Feasibility success rate (%)

	Per iteration		Per algorithm
Data set	Ave	Max	execution
Small	99.6	100.0	100.0
Medium	68.0	100.0	100.0
Large	60.0	100.0	100.0

Comparison with Current Practice
Finally, we present a comparison between the solution found by our heuristic and the solution reported by the firm in a particular case study.

Table 2. Comparison between current practice and proposed heuristic

	NV	NVS	NVD	NT	RC	FC	Total cost
Firm solution	47	9	38	85	$186,018	$70,500	$256,518
Heuristic solution	50	28	22	72	$167,020	$75,000	$242,020

Table 2 shows this comparison itemizing the individual costs and vehicle usage, where NV, NVS, NVD, and NT stand for number of vehicles, single vehicles, double vehicles, and trailers used, respectively. RC and FC are the routing and fixed cost, respectively, and the last column is the total solution cost. The reduction of the heuristic solution is about 6%. It can be seen this is due in great deal to a better arrangement of the single and double vehicles. The new solution uses 3 more vehicles; however, the main difference comes from the number of single- and double-vehicles used. The new solution uses more single-vehicles which yield lower traveling costs overall. It is clear the contrast with the current practice where it was firmly believed that using fewer vehicles (i.e, more double vehicles) would be a better choice.

5 Conclusions

In this paper we studied a vehicle routing problem with pickup and delivery from a real-world application in a bottled-beverage distribution company. Given

the inherent problem difficulty, we have proposed a metaheuristic framework to obtain feasible solutions of good quality. The heuristic and its components were empirically assessed over a wide range of instances. When compared to current practice, it was found that the proposed method obtained solutions of better quality in very reasonable times. The different local search neighborhoods explored in this study can be improved if they can be cast into a more sophisticated local search engine such as tabu search or variable neighborhood search, for instance. This is a subject of follow-up work.

Acknowledgements. This work was improved thanks to the remarks by two anonymous reviewers. This work was supported by the Mexican Council for Science and Technology (CONACYT), grant CB2011-1-166397, and UANL under its Scientific and Technological Research Support Program, grants IT511-10, CE728-11, and HU769-11. The research of the third author has been funded by a scholarship for graduate studies from CONACYT.

References

1. Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G.: Static pickup and delivery problems: A classification scheme and survey. TOP 15(1), 1–31 (2007)
2. Castrillón-Escobar, A.: Una Metaheurística para un Problema de Carga y Descarga en la Repartición de Bebidas Embotelladas. Master thesis, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Mexico (March 2013) (in Spanish)
3. Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., Vogel, U.: Vehicle routing with compartments: Applications, modelling and heuristics. OR Spectrum 33(4), 885–914 (2011)
4. Drexl, M.: Applications of the vehicle routing problem with trailers and transshipments. European Journal of Operational Research 227(2), 275–283 (2013)
5. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6(2), 109–133 (1995)
6. Mitrović-Minić, S., Laporte, G.: Waiting strategies for the dynamic pickup and delivery problem with time windows. Transportation Research 38(7), 635–655 (2004)
7. Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery problems, part II: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft 58(2), 81–117 (2008)
8. Qu, Y., Bard, J.F.: The heterogeneous pickup and delivery problem with configurable vehicle capacity. Transportation Research Part C: Emerging Technologies 32(1), 1–20 (2013)
9. Ropke, S., Cordeau, J.F.: Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science 43(3), 267–286 (2009)
10. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40(4), 455–472 (2006)